MIDI 1.0 Detailed Specification

Document Version 4.2
Revised September 1995

Published by:
The MIDI Manufacturers Association
Los Angeles, CA

This document is a combination of the
MIDI 1.0 Detailed Specification v 4.1.1 and the
MIDI 1.0 Addendum v 4.2
The MIDI Time Code Specification is not included.

Printed 1995

MMA
POB 3173
La Habra CA 90632-3173

TABLE OF CONTENTS

* Specification document available separately (see Table VIII).

APPENDIX

TABLES

INTRODUCTION

MIDI, the Musical Instrument Digital Interface, was established as a hardware and software specification which would make it possible to exchange information (musical notes, program changes, expression control, etc.) between different musical instruments or other devices such as sequencers, computers, lighting controllers, mixers, etc. This ability to transmit and receive data was originally conceived for live performances, although subsequent developments have had enormous impact in recording studios, audio and video production, and composition environments.

This document has been prepared as a joint effort between the MIDI Manufacturers Association (MMA) and the Japan MIDI Standards Committee (JMSC) to explain the MIDI 1.0 specification. This document is subject to change by agreement between the JMSC and MMA. Additional MIDI protocol may be included in supplements to this publication.

HARDWARE

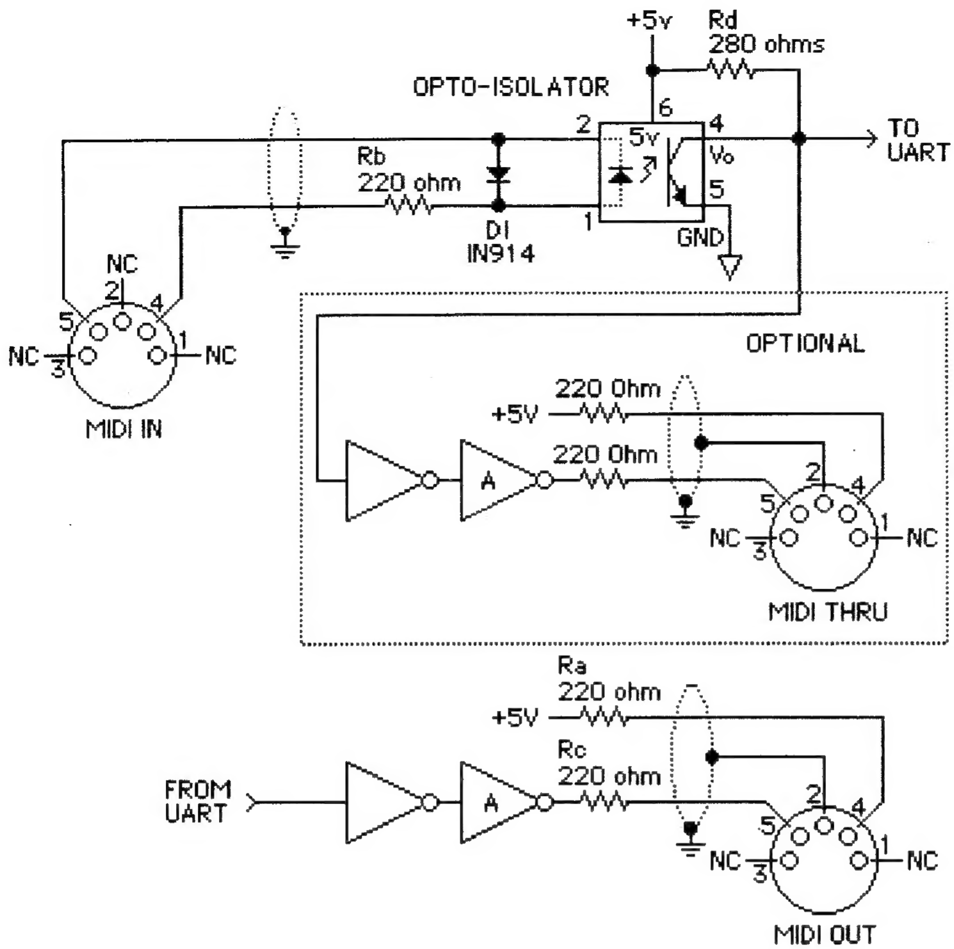
The hardware MIDI interface operates at 31.25 (+/- 1%) Kbaud, asynchronous, with a start bit, 8 data bits (D0 to D7), and a stop bit. This makes a total of 10 bits for a period of 320 microseconds per serial byte. The start bit is a logical 0 (current on) and the stop bit is a logical 1 (current off). Bytes are sent LSB first.

Circuit: *(See Schematic - Page 2)*. 5 mA current loop type. Logical 0 is current ON. One output shall drive one and only one input. To avoid ground loops, and subsequent data errors, the transmitter circuitry and receiver circuitry are internally separated by an opto-isolator (a light emitting diode and a photo sensor which share a single, sealed package). Sharp PC-900 and HP 6N138 opto-isolators have been found acceptable. Other high-speed opto-isolators may be satisfactory. The receiver must require less than 5 mA to turn on. Rise and fall times should be less than 2 microseconds.

Connectors: DIN 5 pin (180 degree) female panel mount receptacle. An example is the SWITCHCRAFT 57 GB5F. The connectors shall be labeled "MIDI IN" and "MIDI OUT". Note that pins 1 and 3 are not used, and should be left unconnected in the receiver and transmitter. Pin 2 of the MIDI In connector should also be left unconnected.

The grounding shield connector on the MIDI jacks should not be connected to any circuit or chassis ground.

When MIDI Thru information is obtained from a MIDI In signal, transmission may occasionally be performed incorrectly due to signal degradation (caused by the response time of the opto-isolator) between the rising and falling edges of the square wave. These timing errors will tend to add up in the "wrong direction" as more devices are chained between MIDI Thru and MIDI In jacks. The result is that, regardless of circuit quality, there is a limit to the number of devices which can be chained (series-connected) in this fashion.



MIDI Standard Hardware

NOTES:

1. Opto-isolator currently shown is Sharp PC-900
   (HP 6N138 or other opto-isolator can be used with appropriate changes.)

2. Gates "A" are IC or transistor.

3. Resistors are 5%

Cables shall have a maximum length of fifty feet (15 meters), and shall be terminated on each end by a corresponding 5-pin DIN male plug, such as the SWITCHCRAFT 05GM5M. The cable shall be shielded twisted pair, with the shield connected to pin 2 at both ends.

A MIDI Thru output may be provided if needed, which provides a direct copy of data coming in MIDI In. For long chain lengths (more than three instruments), higher-speed opto-isolators should help to avoid additive rise/fall time errors which affect pulse width duty cycle.

DATA FORMAT

MIDI communication is achieved through multi-byte "messages" consisting of one Status byte followed by one or two Data bytes. Real-Time and Exclusive messages are exception.

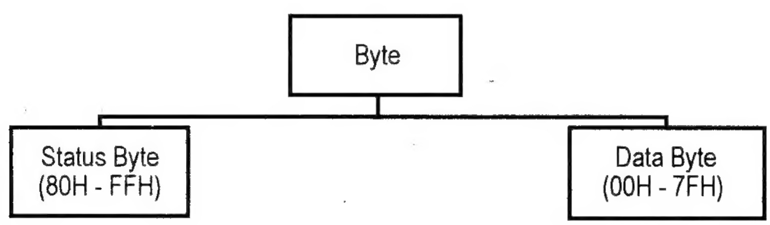
A MIDI-equipped instrument typically contains a receiver and a transmitter. Some instruments may contain only a receiver or only a transmitter. A receiver accepts messages in MIDI format and executes MIDI commands. It consists of an opto-isolator, Universal Asynchronous Receiver/Transmitter (UART), and any other hardware needed to perform the intended functions. A transmitter originates messages in MIDI format, and transmits them by way of a UART and line driver.

MIDI makes it possible for a user of MIDI-compatible equipment to expand the number of instruments in a music system and to change system configurations to meet changing requirements.

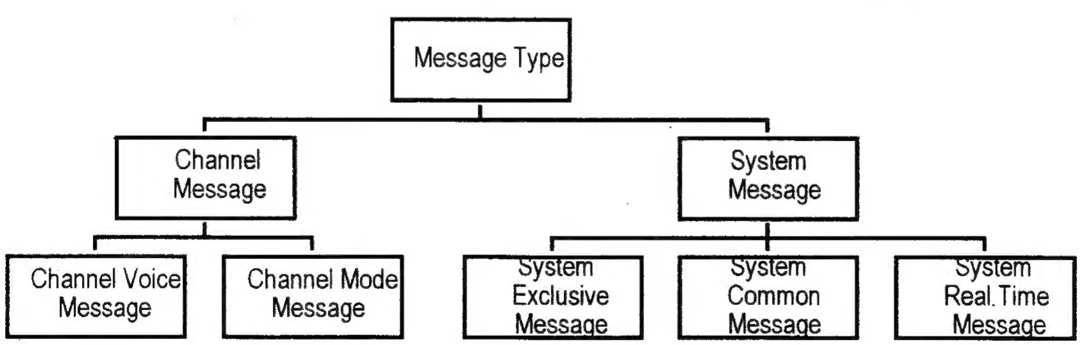
MIDI messages are sent over any of 16 channels which are used for a variety of performance information. There are five major types of MIDI messages: Channel Voice, Channel Mode, System Common, System Real-Time and System Exclusive.

A MIDI event is transmitted as a "message" and consists of one or more bytes. The diagrams below show the structure and classification of MIDI data.

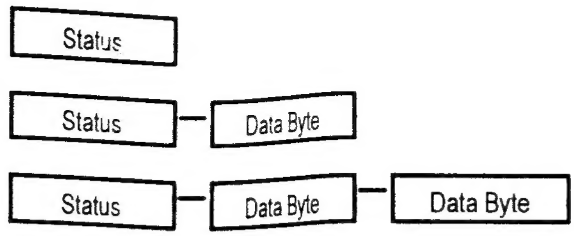
TYPES OF MIDI BYTES:



TYPES OF MIDI MESSAGES:



STRUCTURE OF A SINGLE MESSAGE:



STRUCTURE OF SYSTEM EXCLUSIVE MESSAGES:

MESSAGE TYPES

Messages are divided into two main categories: *Channel* and *System*.

CHANNEL MESSAGES

A Channel message uses four bits in the Status byte to address the message to one of sixteen MIDI channels and four bits to define the message (see Table II). Channel messages are thereby intended for the receivers in a system whose channel number matches the channel number encoded into the Status byte.

An instrument can receive MIDI messages on more than one channel. The channel in which it receives its main instructions, such as which program number to be on and what mode to be in, is referred to as its "Basic Channel". An instrument may be set up to receive performance data on multiple channels (including the Basic Channel). These are referred to as "Voice Channels". These multiple-channel situations will be discussed in more detail later.

There are two types of Channel messages: *Voice* and *Mode*.

VOICE: To control an instrument's voices. Voice messages are sent over the Voice Channels.

MODE: To define the instrument's response to Voice messages. Mode messages are sent over an instrument's Basic Channel.

SYSTEM MESSAGES

System messages are not encoded with channel numbers. There are three types of System messages: *Common*, *Real-Time*, and *Exclusive*.

COMMON: Common messages are intended for all receivers in a system regardless of channel.

REAL-TIME: Real-Time messages are used for synchronization and are intended for all clock-based units in a system. They contain Status bytes only — no Data bytes. Real-Time messages may be sent at any time — even between bytes of a message which has a different status. In such cases the Real-Time message is either acted upon or ignored, after which the receiving process resumes under the previous status.

EXCLUSIVE: Exclusive messages can contain any number of Data bytes, and can be terminated either by an End of Exclusive (EOX) or any other Status byte (except Real Time messages). An EOX should always be sent at the end of a System Exclusive message. These messages include a Manufacturer's Identification (ID) code. If a receiver does not recognize the ID code, it should ignore the following data.

So that other users and third party developers can fully access their instruments, manufacturers must publish the format of the System Exclusive data following their ID code. Only the manufacturer can define or update the format following their ID.

DATA TYPES

There are two types of bytes sent over MIDI: *Status Bytes* and *Data bytes*.

STATUS BYTES

Status bytes are eight-bit binary numbers in which the Most Significant Bit (MSB) is set (binary 1). Status bytes serve to identify the message type, that is, the purpose of the Data bytes which follow it. Except for Real-Time messages, new Status bytes will always command a receiver to adopt a new status, even if the last message was not completed.

RUNNING STATUS

For Voice and Mode messages only. When a Status byte is received and processed, the receiver will remain in that status until a different Status byte is received. Therefore, if the same Status byte would be repeated, it can optionally be omitted so that only the Data bytes need to be sent. Thus, with Running Status, a complete message can consist of only Data bytes.

Running Status is especially helpful when sending long strings of Note On/Off messages, where "Note On with Velocity of 0" is used for Note Off.

Running Status will be stopped when any other Status byte intervenes. Real-Time messages should not affect Running Status.

See also: Additional Explanations and Application Notes

UNIMPLEMENTED STATUS

Any status bytes, and subsequent data bytes, received for functions not implemented in a receiver should be ignored.

UNDEFINED STATUS

All MIDI devices should be careful to never send any undefined status bytes. If a device receives any such code, it should be ignored without causing any problems to the system. Care should also be taken during power-up and power-down that no messages be sent out the MIDI Out port. Such noise, if it appears on a MIDI line, could cause a data or framing error if the number of bits in the byte are incorrect.

DATA BYTES

Following a Status byte (except for Real-Time messages) there are either one or two Data bytes which carry the content of the message. Data bytes are eight-bit binary numbers in which the Most Significant Bit (MSB) is always set to binary 0. The number and range of Data bytes which must follow each Status byte are specified in the tables in section 2. For each Status byte the correct number of Data bytes must always be sent. Inside a receiver, action on the message should wait until all Data bytes required under the current status are received. Receivers should ignore Data bytes which have not been properly preceded by a valid Status byte (with the exception of "Running Status," explained above).

CHANNEL MODES

Synthesizers and other instruments contain sound generation elements called voices. Voice assignment is the algorithmic process of routing Note On/Off data from incoming MIDI messages to the voices so that notes are correctly sounded.

Note: When we refer to an "instrument" please note that one physical instrument may act as several virtual instruments (i.e. a synthesizer set to a 'split' mode operates like two individual instruments). Here, "instrument" refers to a virtual instrument and not necessarily one physical instrument.

Four Mode messages are available for defining the relationship between the sixteen MIDI channels and the instrument's voice assignment. The four modes are determined by the properties Omni (On/Off), Poly, and Mono. Poly and Mono are mutually exclusive, i.e., Poly disables Mono, and vice versa. Omni, when on, enables the receiver to receive Voice messages on all voice Channels. When Omni is off, the receiver will accept Voice messages from only selected Voice Channel(s). Mono, when on, restricts the assignment of Voices to just one voice per Voice Channel (Monophonic.) When Mono is off (Poly On), a number of voices may be allocated by the Receiver's normal voice assignment (Polyphonic) algorithm.

For a receiver assigned to Basic Channel "N," (1-16) the four possible modes arising from the two Mode messages are:

Mode	Omni		
1	On	Poly	Voice messages are received from all Voice channels and assigned to voices polyphonically.
2	On	Mono	Voice messages are received from all Voice Channels, and control only one voice, monophonically.
3	Off	Poly	Voice messages are received in Voice channel N only, and are assigned to voices polyphonically.
4	Off	Mono	Voice messages are received in Voice channels N through N+M-1, and assigned monophonically to voices 1 through M, respectively. The number of voices "M" is specified by the third byte of the Mono Mode Message.

Four modes are applied to transmitters (also assigned to Basic Channel N). Transmitters with no channel selection capability should transmit on Basic Channel 1 (N=1).

Mode	Omni		
1	On	Poly	All voice messages are transmitted in Channel N.
2	On	Mono	Voice messages for one voice are sent in Channel N.
3	Off	Poly	Voice messages for all voices are sent in Channel N.
4	Off	Mono	Voice messages for voices 1 through M are transmitted in Voice Channels N through N+M-1, respectively. (Single voice per channel).

A MIDI receiver or transmitter operates under only one Channel Mode at a time. If a mode is not implemented on the receiver, it should ignore the message (and any subsequent data bytes), or switch to an alternate mode, usually Mode 1 (Omni On/Poly).

Mode messages will be recognized by a receiver only when received in the instrument's Basic Channel — regardless of which mode the receiver is currently assigned to. Voice messages may be received in the Basic Channel and in other Voice Channels, according to the above specifications.

Since a single instrument may function as multiple "virtual" instruments, it can thus have more than one basic channel. Such an instrument behaves as though it is more than one receiver, and each receiver can be set to a different Basic Channel. Each of these receivers may also be set to a different mode, either by front panel controls or by Mode messages received over MIDI on each basic channel. Although not a true MIDI mode, instruments operating in this fashion are described as functioning in "Multi Mode."

An instrument's transmitter and receiver may be set to different modes. For example, an instrument may receive in Mono mode and transmit in Poly mode. It is also possible to transmit and receive on different channels. For example, an instrument may receive on Channel 1 and transmit on Channel 3.

POWER-UP DEFAULT CONDITIONS

It is recommended that at power-up, the basic channel should be set to 1, and the mode set to Omni On/Poly (Mode 1). This, and any other default conditions for the particular instrument, should be maintained indefinitely (even when powered down) until instrument panel controls are operated or MIDI data is received. However, the decision to implement the above, is left totally up to the designer.

CHANNEL VOICE MESSAGES

Note-Off	8nH	
Note-On	9nH	
Poly Key Pressure	AnH	
Control Change	BnH	(0 - 119)
Program Change	CnH	
Channel Pressure	DnH	
Pitch Bend	EnH	

Channel Voice Messages are the bulk of information transmitted between MIDI instruments. They include all Note-On, Note-Off, program change, pitch-wheel change, after-touch pressure and controller changes. These terms are defined below.

A single Note-On message consists of 3 bytes, requiring 960 microseconds for transmission. When many notes are played at the same time, the multiple Note-On messages may take several milliseconds to transmit. This can make it difficult for MIDI to respond to a large number of simultaneous events without some slight audible delay. This problem can be relieved to some degree by using the Running Status mode described on page 5 and in the appendix (A1-3).

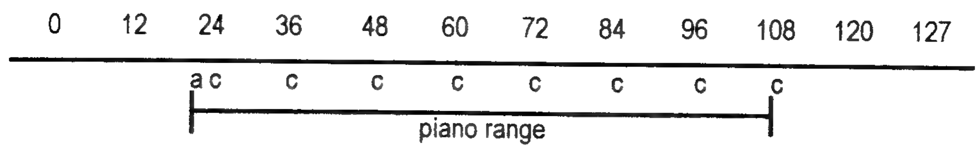
TYPES OF VOICE MESSAGES

NOTE-ON:	Message is sent by pressing a key or from other triggering devices.
NOTE-OFF:	Message is sent by releasing a key.
CONTROL CHANGE:	Message is sent when a controller other than a key (e.g. a pedal, wheel, lever, switch, etc.) is moved in order to modify the sound of a note (e.g. introducing modulation, sustain, etc.). Control changes are not used for sending parameters of tones (voices), such as attack time, filter cut off frequency, etc.
PROGRAM CHANGE:	When a "program" (i.e. sound, voice, tone, preset or patch) is changed, the number corresponding to the newly selected program is transmitted.
AFTER TOUCH:	This message typically is sent by key after-pressure and is used to modify the note being played. After touch messages can be sent as Polyphonic Key Pressure or Channel Pressure.
PITCH BEND CHANGE:	This message is used for altering pitch. The maximum resolution possible is 14 bits, or two data bytes.

Voice messages are not exclusively for use by keyboard instruments, and may be transmitted for a variety of musical purposes. For example, Note-On messages generated with a conventional keyboard synthesizer may be used to trigger a percussion synthesizer or lighting controller.

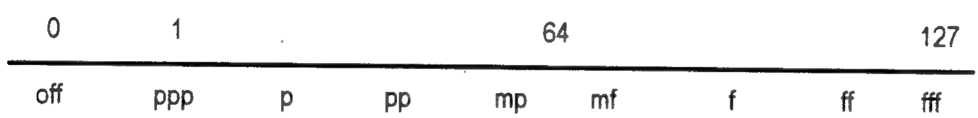
NOTE NUMBER

Each note is assigned a numeric value, which is transmitted with any Note-On/Off message. Middle C has a reference value of 60. This is the middle C of an 88 note piano-style keyboard though it need not be physically located in the center of a keyboard.



VELOCITY

Interpretation of the Velocity byte is left up to the receiving instrument. Generally, the larger the numeric value of the message, the stronger the velocity-controlled effect. If velocity is applied to volume (output level) for instance, then higher Velocity values will generate louder notes. A value of 64 (40H) would correspond to a mezzo-forte note and should also be used by device without velocity sensitivity. Preferably, application of velocity to volume should be an exponential function. This is the suggested default action; note that an instrument may have multiple tables for mapping MIDI velocity to internal velocity response.



vvvvvvv = 64: if not velocity sensitive
vvvvvvv = 0: Note-Off (with velocity of 64)

NOTE-OFF

MIDI provides two roughly equivalent means of turning off a note (voice). A note may be turned off either by sending a Note-Off message for the same note number and channel, or by sending a Note-On message for that note and channel with a velocity value of zero. The advantage to using "Note-On at zero velocity" is that it can avoid sending additional status bytes when Running Status is employed.

Due to this efficiency, sending Note-On messages with velocity values of zero is the most commonly used method. However, some keyboard instruments implement release velocity where a Note-Off code (8nH) accompanied by a "velocity off" byte is used. A receiver must be capable of recognizing either method of turning off a note, and should treat them identically.

The three methods of using Note-On (9nH) or Note-Off (8nH) are as follows:

1. For a keyboard which does not implement Velocity, the note will be turned on using 9n, kkkkkkk, 64 (40H) and may be turned off using 9n, 0kkkkkkk, 00000000 or 8n, 0kkkkkkk, 0xxxxxxx (a value of 64 [40H] is used for x).

2. For a keyboard which incorporates Key On Velocity, but not Release Velocity the note is turned on using 9n 0kkkkkkk. 0vvvvvvv and may be turned off using 9n. 0kkkkkkk. 00000000 or 8n. 0kkkkkkk. 0xxxxxxx (a value of 64 (40H) is recommended for x).

3. Where the keyboard implements both Key On Velocity and Release Velocity, a note is turned on using 9n 0kkkkkkk. 0vvvvvvv, and turned off using 8n. 0kkkkkkk, 0vvvvvvv.

CONTROL CHANGE

The Control Change message is generally used for modifying tones with a controller other than a keyboard key. It is not for setting synthesizer parameters such as VCF cut-off, envelope decay, etc. There are some exceptions to the use of the Control Change message, such as the special Bank Select message and the RPN/NRPN messages (listed below).

CONTROLLER NUMBERS

All controller number assignments are designated by agreement between the MMA and JMSC. The numbers listed in Table III are specified for standard musical instrument applications. However, many non-musical devices which implement MIDI, such as lighting controllers, may use designated controller numbers at their discretion. Due to the limited number of controller numbers it would be impossible to assign a number to every possible effect (musical and non-musical) used now and in the future. For this reason, controllers are generally assigned only for purposes associated with musical instruments.

It is up to the manufacturer to inform their users of the fact that a device is using non-standard controller assignments. Though controllers may be used for non-musical applications, they must still be used in the format detailed in Table II. Manufacturers can request through the MMA or JMSC that logical controllers be assigned to physical ones as needed. A controller allocation table should be provided in the user's operation manual of all products.

A manufacturer wishing to control a number of device-specific parameters over MIDI should used non-registered parameter numbers and the Data Entry controllers (Data Entry Slider, Increment, and Decrement messages) as opposed to a large number of controllers. This alleviates possible conflict with devices responding to the same control numbers unpredictably.

There are currently 120 controller numbers, from 0 through 119 (controller 120 was recently adopted as a Channel Mode Message and is no-longer considered a Control Change). As shown below, controller numbers 32 to 63 are used to define an LSB byte for corresponding controllers 0 through 31. Controller classifications are as follows:

0	through	31	=	MSB of most continuous Controller Data
32	through	63	=	LSB for controllers 0 through 31
64	through	95	=	Additional single-byte controllers
96	through	101	=	Increment/Decrement and Parameter numbers
102	through	119	=	Undefined single-byte controllers

A numeric value (controller number) is assigned to the controllers of the transmitting instrument. A receiver may use the message associated with a controller number to perform any operation or achieve any desired effect. Further, a single controller number may be used to change a number of parameters. controller numbers are classified by various categories. Each controller number corresponds to one byte of data.

Controller numbers 0 through 31 are for controllers that obtain information from pedals, levers, wheels, etc. Controller numbers 32 through 63 are reserved for optional use as the LSB (Least Significant Byte) when higher resolution is required and correspond to 0 through 31 respectively. For example, controller number 7 (Volume) can represent 128 steps or increments of some controller's position. If controller number 39, the corresponding LSB number to controller number 7, is also used, 14-bit resolution is obtained. This provides for resolution of 16,384 steps instead of 128.

If 128 steps of resolution is sufficient the second byte (LSB) of the data value can be omitted. If both the MSB and LSB are sent initially, a subsequent fine adjustment only requires the sending of the LSB. The MSB does not have to be retransmitted. If a subsequent major adjustment is necessary the MSB must be transmitted again. When an MSB is received, the receiver should set its concept of the LSB to zero.

All controller numbers 64 and above have single-byte values only, with no corresponding LSB. Of these, 64 through 69 have been defined for switched functions (hold pedal, etc.) while 91 through 95 are for controlling the depth of certain external audio effects.

Control numbers 64 through 69 are assigned to functions normally associated with switches (i.e. sustain or soft pedals). However these controllers can be used to send any continuous value. The reverse can also be true for a continuous controller such as Modulation Wheel. While this controller is most often used as a variable control, an on/off modulation switch can also be used. This would be accomplished by sending the Modulation Controller number (01) and a data byte of either 0 (off) or 127 (on).

If a receiver is expecting switch information it should recognize 0-63 (00H-3FH) as "OFF" and 64-127 (40H-7FH) as "ON". This is because a receiver has no way of knowing whether the message information is from a switch or a continuous controller. It is very important to always use an existing control number. The control numbers already adopted for use are listed in Table III. We will discuss some of them, but not all, below.

GLOBAL CONTROLLERS

If a receiving instrument is in Mode 4 (Omni Off/Mono) and is thus able to respond to more than one MIDI channel, it is possible to use a Global Controller to affect all voices regardless of MIDI channel. This is accomplished by sending any controller intended to affect all voices over the MIDI channel one below the basic channel of the receiver. For example, if a receiving synthesizer in Mode 4 is responding to channels 6 through 12, its basic channel is 6. Any controllers received on channel 5 would be Global Controllers and would affect all voices. If the Basic Channel is 1, then the Global Channel wraps to become 16, though not all receivers may provide this function.

GENERAL PURPOSE CONTROLLERS

Controller numbers 16-19 and 80-83 are defined as General Purpose Controllers. They may be used by a manufacturer for any added functions able to send or receive some sort of control information needed for a specific product. They do not have any intrinsic functions assigned to them. General Purpose Controllers 16-19 are two byte controllers (with controller numbers 48-51 for an optional LSB). General Purpose Controllers 80-83 are single byte controllers. As an example, an instrument with a special, user definable joystick or lever assignable to any internal parameter could send and receive General Purpose Controller numbers for sequencing.

CONTROLLER EFFECT

All transmitters should send a value of 00 to represent minimum and 127 (7FH) to represent maximum. For continuous controllers without a center detented position, it is recommended that the minimum effect position correspond to 00, and the maximum effect position correspond to 127 (7FH).

Virtually all controllers are defined as 0 being no effect and 127 being maximum effect. There are three defined controllers that are notably different: Balance, Pan and Expression.

BALANCE:	A Balance Controller has been adopted as continuous controller number 8 (08H) with value 00 = full volume for the left or lower half, 64 (40H) = equal balance, and 127 (7FH) = full volume for the right or upper half. This controller determines the volume balance between two different sound sources.
PAN:	A Pan Controller has been adopted as continuous controller number 10 (0AH) with value 00 = hard left, 64 (40H) = center, and 127 (7FH) = hard right. This controller determines where a single sound source will be located in a stereo field.
EXPRESSION:	An Expression Controller has been adopted as continuous controller number 11 (0BH). Expression is a form of volume accent above the programmed or main volume.

BANK SELECT

Bank Select is a special controller. The Bank Select message is an extension of the Program Change message and is used to switch between multiple banks. For example, a bank select message could be used to select more than 128 programs, or switch between internal memory and external RAM card.

Control Change numbers 00H and 20H are defined as the Bank Select message. 00H is the MSB and 20H is the LSB for a total of 14 bits. This allows 16,384 banks to be specified.

The transmitter must transmit the MSB and LSB as a pair, and the Program Change must be sent immediately after the Bank Select pair. If their is any delay between these messages and they are passed through a merging device (which may insert another message) the message may be interpreted incorrectly.

The messages Bank Select MSB, LSB and Program number will select a specific program. After switching to another bank, any Program Change messages transmitted singularly will select other program in that bank.

After the receiver as received the entire Bank Select messages it will normally change to a new program. The program must change upon the receipt of the Program Change message. However, the program need not be changed for a note which is already sounding. When the Bank Select message is received, the receiving device must remember that bank number in readiness for the following Program Change. Bank Select alone must not change the program. This is to assure that multiple devices change concurrently.

The 14 bit Bank Select value corresponds to bank numbers as follows:

MSB	LSB	Bank Number
00H	00H	Bank 1
00H	7FH	Bank 128
01H	00H	Bank 129
7FH	7FH	Bank 16,384

As with program numbers, banks begin counting from 1. Thus the actual bank number will be (MIDI value + 1).

LEGATO FOOTSWITCH

Bn 44 vv	Legato Footswitch	
	vv = 00-3F	Normal
	vv = 40-7F	Legato

Legato Footswitch is a recent addition to the specification. This controller is used to turn monophonic legato response in a receiving instrument on and off. When turned on the instrument goes into a monophonic mode: if a new Note-On is received before the Note-Off for the currently sounding note, pitch is changed without re-attacking the envelopes or (if possible) playing the attack portion of the sound. When turned off the voice assignment mode (polyphonic or monophonic) returns to the state it was in prior to receiving the Legato On command.

Note: This message is not a replacement for proper Mode 4 legato operation. Nor is it a replacement for sending Note-Offs for every Note-On sent. It is specifically intended as a useful performance controller.

EFFECTS CONTROLLER REDEFINITION

Controller numbers 91 – 95 are defined as Effects Depth 1 through Effects Depth 5 and can be used for controlling various effects. Their former titles of External Effects Depth, Tremolo Depth, Chorus Depth, Celeste (Detune) Depth, and Phaser Depth are now the recommended defaults.

SOUND CONTROLLERS

Controllers 46H through 4FH are defined as "Sound Controllers." Manufacturers and users may map any functions they desire to these ten controllers. However, to further aid standardization and easy set-up for users, the MMA and JMSC may determine "default" assignments for these controllers. A manufacturer may independently assign other functions to these controllers, but it should be understood that the MMA and JMSC may later assign different defaults to them.

Five Sound Controller defaults have currently been defined by the MMA and JMSC:

Number	Name	Instruments
46H (70)	Sound Controller #1	Sound Variation
47H (71)	Sound Controller #2	Timber/Harmonic Intensity
48H (72)	Sound Controller #3	Release Time
49H (73)	Sound Controller #4	Attack Time
4AH (74)	Sound Controller #5	Brightness

SOUND VARIATION CONTROLLER:

`Bn 46 vv`	Sound Variation

This controller is used to select alternate versions of a sound during performance. Note that it is different from a program change in several ways:

1. The variation (alternate sound) is an intrinsic part of the program which is being played, and is programmed in the patch.

2. The variation is usually related to the primary sound – for example, a sax and an overblown sax, bowed and pizzicato strings, a strummed and muted guitar, etc.

3. The variation to be used is decided at the time of the Note-On. For example, if the value of SVC is set to 00, notes are sounded, and then SVC is changed to 24H, the notes currently sounding will not change. Any new notes will take the variation determined by the new SVC value. If the old notes are released, they will finish in their original manner.

SVC actually acts as a multi-level switch. An instrument's levels of variations should be mapped over the entire 00-7FH range of the controller. For example, if an instrument had only a single SVC switch, it would transmit a value of 00) for the primary sound and 7FH for the secondary sound. If an instrument had four variations, it would transmit these as 00, 20H, 40H, and 7FH. The first instrument would receive any value in the range of 40H-7FH to select its secondary sound.

TIMBRE CONTROLLERS:

`Bn 47 vv`	Timbre/Harmonic Intensity
`Bn 4A vv`	Brightness

The Harmonic Content controller (commonly known as "timbre") is intended as a modifier of the harmonic content of a sound, e.g. FM feedback, FM modulation amount, waveform loop points, etc. The receiving instrument determines the application of this controller according to its voice architecture.

Harmonic Content should be treated as an absolute rather than relative value, and should be handled as a modulation input by the receiver.

The Brightness controller is aimed specifically at altering the brightness of the sound. In most sound modules, it would correspond to a low pass filter's cutoff frequency, although it might also control EQ or a harmonic enhancer/exciter.

Brightness should be treated as a relative controller, with a data value of 40H meaning no change, values less than 40H meaning progressively less bright, and values greater than 40H meaning progressively more bright.

Both of these controllers are intended as a performance controllers – not as a sound parameter editing controllers (in other words, these messages do not change the memorized data of a preset). They have no fixed association with any physical controller.

The receiving instrument should be able to respond to these controllers while sustaining notes without audible glitches or re-triggering of the sound. The effective range of these controllers may be programmed per preset, if desired.

ENVELOPE TIME CONTROLLERS:

`Bn 48 vv`	Release Time	
`Bn 49 vv`	Attack Time	
	`vv = 00 - 3F`	*= shorter times (00 = shortest adjustment)*
	`vv = 40`	*= no change in times*
	`vv = 41 - 7F`	*= longer times (7F = longest adjustment)*

These controllers are intended to adjust the attack and release times of a sound relative to its pre-programmed values. The manufacturer and user should decide which envelopes in a voice are affected: the default should be all envelopes. These controllers should affect all envelopes affected about to enter their release or attack phases (respectively): the manufacturer may allow an option to affect envelope phases already started.

These envelope time controllers do not replace the effect attack or release velocity may have on the envelope times of the sound: they should interact with them in a predictable manner. They have no fixed association with a physical controller. The effective range of these controllers may be programmed per preset. if desired.

These are intended as a performance controllers – not as a sound parameter editing controllers (in other words. these messages do not change the memorized data of a preset). The receiving instrument should be able to respond to these controllers while sustaining notes and without audible glitches or re-triggering of the sound.

PORTAMENTO CONTROLLER

`Bn 54 kk`

`n`	= channel
`kk`	= source note number for pitch reference

Portamento Control (PTC) is a recent addition. and defines a continuous controller that communicates which note number the subsequent note is gliding from. It is intended for special effects in playing back pre-sequenced material. so that legato with portamento may be realized while in Poly mode.

When a Note-On is received after a Portamento Control message. the voice's pitch will glide from the key specified in the Portamento Control message to the new Note-On's pitch at the rate set by the portamento time controller (ignoring portamento on/off).

A single Portamento Control message only affects the next Note-On received on the matching channel (i.e. it is reset after the Note-On). Receiving a Portamento Control message does not affect the pitch of any currently sustaining or releasing notes in Poly modes; if in Mono mode or if Legato Footswitch is currently on. a new overlapped note event would result in an immediate pitch jump to the key specified by the portamento Control message, and then a glide at the current portamento rate to the key specified by the new Note-On.

In all modes. the note is turned off by a note that matches the Note-On key number; not the key number stated in the Portamento Control message. Pitch bend offsets the pitch used by both the Portamento Control starting note and the target Note-On.

If there is a currently sounding voice whose note number is coincident with the source note number. the voice's pitch will glide to the new Note-On's pitch according to the portamento time without re-attacking. Then. no new voice should be assigned.

The single Portamento Control message only affects the next Note-On received on the matching channel (in other words, it is reset after the Note-On). Receiving a Portamento Control message does not affect the pitch of other currently sounding voices except a voice whose note number is coincident with the source key number of the Portamento Control message in Poly mode.

Example 1:

MIDI Message	Description	Result
`90 3C 40`	Note-On #60	#60 on (middle C)
`B0 54 3C`	PTC from #60	no change in current note
`90 40 40`	Note-On #64	re-tune from #60 to #64
`80 3C 40`	Note-Off #60	no change
`80 40 40`	Note-Off #64	#64 off

Example 2:

MIDI Message	Description	Result
`B0 54 3C`	PTC from #60	no change
`90 40 40`	Note-On #64	#64 with glide from #60
`80 40 40`	Note-Off #64	#64 Off

REGISTERED AND NON-REGISTERED PARAMETER NUMBERS

Registered and Non-Registered Parameter Numbers are used to represent sound or performance parameters. As noted below, Registered Parameters Numbers are agreed upon by the MMA and JMSC. Non-Registered Parameter Numbers may be assigned as needed by individual manufacturers. The basic procedure for altering a parameter value is to first send the Registered or Non-Registered Parameter Number corresponding to the parameter to be modified, followed by the Data Entry, Data Increment, or Data Decrement value to be applied to the parameter.

There are several rules and suggestions as to the use of these parameter numbers and controllers:

1. A manufacturer may assign any desired parameter to any Non-Registered Parameter Number. This list should be published in the owner's manual.

2. Reception of Non-Registered Parameter Numbers should be disabled on power-up to avoid confusion between different machines. Transmission of these numbers should be safe at any time if this is done.

3. After the reception of Non-Registered (or Registered) Parameter Numbers has been enabled, the receiver should wait until it receives both the LSB and MSB for a parameter number to ensure that it is operating on the correct parameter.

4. The receiver should be able to respond accordingly if the transmitter sends only an LSB or MSB to change the parameter number. However, since the transmitter can't know when reception was enabled on the receiver which will be waiting for both the LSB and MSB (at least initially), it is recommended that the LSB and MSB be sent each time a new parameter number is selected.

5. The Registered Parameter Numbers are agreed upon by the MMA and JMSC. Since this is a standardized list, reception of these Registered Parameter Numbers may be enabled on power-up.

6. Once a new Parameter Number is chosen, that parameter retains its old value until a new Data Entry, Data Increment, or Data Decrement is received.

PITCH BEND SENSITIVITY:

Pitch Bend Sensitivity is defined as Registered Parameter Number 00 00. The MSB of Data Entry represents the sensitivity in semitones and the LSB of Data Entry represents the sensitivity in cents. For example, a value of MSB=01, LSB=00 means +/- one semitone (a total range of two semitones).

MASTER TUNING:

Registered Parameter numbers 01 and 02 are used for Master Tuning control. They are implemented as follows:

RPN 01 - FINE TUNING:

Resolution:	100/8192 cents
Range:	100/8192* (-8192) to 100/8192* (+8191)

Control MSB	Value LSB	Displacement in cents from A440
00	00	100/8192* (-8192)
40H	00	100/8192* (0)
7FH	7FH	100/8192* (+8191)

RPN 02 - COARSE TUNING:

Resolution:	100 cents
Range:	100* (-64) to 100* (+63)

Control MSB	Value LSB	Displacement in cents from A440
00	XX	100* (-64)
40H	XX	100* (0)
7FH	XX	100* (+63)

PROGRAM CHANGE

This message is used to transmit the program or "patch" number when changing sounds on a MIDI instrument. The message does not include any information about the sound parameters of the selected tone. As the various parameters that constitute a program are very different from one MIDI instrument to another it is much more efficient to address a sound simply by its internal number.

Program Change messages are most often sent when physically selecting a new sound on an instrument. However, if the transmitting instrument does not produce its own sound, a button or any other physical controller can be used for transmitting program change messages to slave devices.

It is not often that the exact same tones are in the transmitting and receiving instruments, so some care must be taken when assigning tones to a given tone number. The ability to reassign programs to a given program change number should be part of an instrument's capabilities. Some instruments number their internal patches in octal numerics. This should have no effect on the numbers used for patch change. Numbering should begin with 00H and increment sequentially. For example, octal 11 would be 00H, 12 would be 01H, etc.

It may not always be desirable for a tone change in a transmitting instrument to cause a program change in a receiving instrument. Some means of disabling the sending or reception of program change should be provided. Program change messages do not necessarily need to change tones. In some instruments, such as a drum machine, the message may be used to switch to a different rhythmic pattern. In MIDI controlled effects devices, the program change message may be used to select a different preset effect.

Note: also see Bank Select.

PITCH BEND CHANGE

This function is a special purpose pitch change controller, and messages are always sent with 14 bit resolution (2 bytes). In contrast to other MIDI functions, which may send either the LSB or MSB, the Pitch Bender message is always transmitted with both data bytes. This takes into account human hearing which is particularly sensitive to pitch changes. The Pitch Bend Change message consists of 3 bytes when the leading status byte is also transmitted. The maximum negative swing is achieved with data byte values of 00, 00. The center (no effect) position is achieved with data byte values of 00, 64 (00H, 40H). The maximum positive swing is achieved with data byte values of 127, 127 (7FH, 7FH).

Sensitivity of Pitch Bend Change is selected in the receiver. It can also be set by the receiver or transmitted via Registered Parameter number 00 00.

AFTERTOUCH

Two types of Aftertouch messages are available: one that affects an entire MIDI channel and one that affects each individual note played. They are differentiated by their status byte. In either case, the Aftertouch value is determined by horizontally moving the key (front-to-rear or left-to-right), or by pressing down on the key after it "bottoms out". Devices such as wind controllers can send Aftertouch from increasing breath pressure after the initial attack. The type of tone modification created by the Aftertouch is determined by the receiver. Aftertouch may be assigned to affect volume, timbre, vibrato, etc.

If a "Channel Pressure" (Dn, 0vvvvvvv) message is sent, then the Aftertouch will affect all notes playing in that channel.

If a "Polyphonic Key Pressure" (An, 0kkkkkkk, 0vvvvvvv) message is sent discrete Aftertouch is applied to each note (0kkkkkkk) individually.

CHANNEL MODE MESSAGES

(Control Change Status)	BnH
All Sound Off	120
Reset All Controllers	121
Local Control	122
All Notes Off	123
Omni Off	124
Omni On	125
Mono On (Poly Off)	126
Poly On (Mono Off)	127

A Mode message is sent with the same Status Byte as a Control Change message. The second byte of the message will be between 121 (79H) and 127 (7FH) to signify a mode message. Mode messages determine how an instrument will receive all subsequent voice messages. This includes whether the receiver will play notes monophonically or polyphonically and whether it will respond only to data sent on one specific voice channel or all of them.

MODE MESSAGES AS ALL NOTES OFF MESSAGES

Messages 123 through 127 also function as All Notes Off messages. They will turn off all voices controlled by the assigned Basic Channel. These messages should not be sent periodically, but only for a specific purpose. In no case should they be used in lieu of Note Off commands to turn off notes which have been previously turned on. Any All Notes Off command (123-127) may be ignored by a receiver with no possibility of notes staying on, since any Note On command must have a corresponding specific Note Off command.

THE BASIC CHANNEL OF AN INSTRUMENT

Mode messages are recognized only when sent on the Basic Channel to which the receiver is assigned, regardless of the current mode. The Basic Channel is set in the transmitter or receiver either by permanent "hard wiring," by panel controls, or by System Exclusive messages, and cannot be changed by any MIDI mode or voice message. Mode messages can only be transmitted and received on an instrument's Basic Channel.

RECEIVER'S MODE (OMNI ON/OFF & POLY/MONO)

The receiver can be set to any of four modes which determine how it will recognize voice messages. The four modes are set with two mode messages: Omni On/Off, and Poly/Mono.

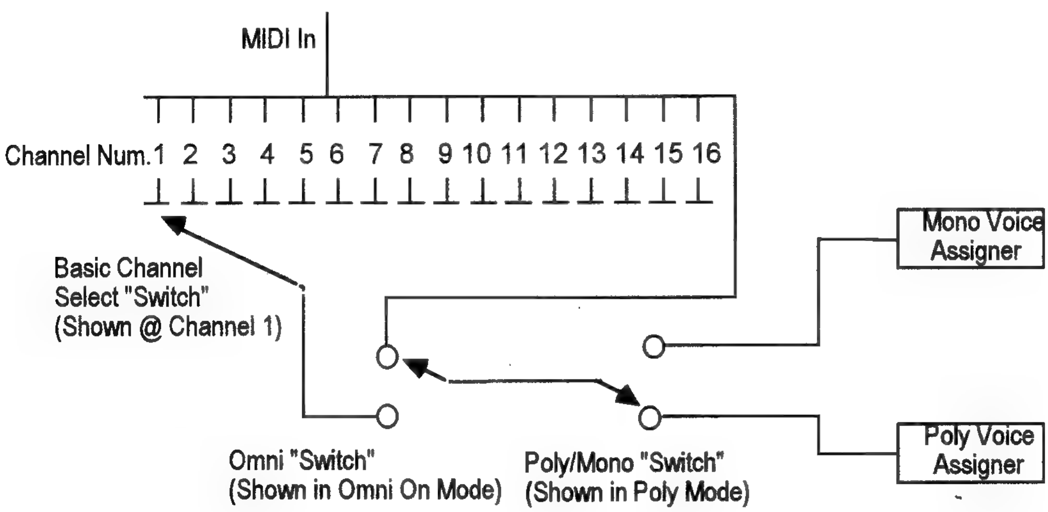
Mode 1:	Omni On, Poly
Mode 2:	Omni On, Mono
Mode 3:	Omni Off, Poly
Mode 4:	Omni Off, Mono

Mono and Poly determine how the receiver's voices will be assigned when more than one note is received simultaneously. In Mono mode, each voice in the receiver will respond monophonically to note messages on a particular MIDI channel. This would be like having several Monophonic synthesizers in a single box. In Poly mode, voices in the receiver will respond to note messages polyphonically.

These four modes may be changed by panel controls on the receiver. Care must be taken, however, since it is possible that the receiver may be "disabled" by setting it to a mode where it will not recognize or correctly respond to data received from a transmitter. As the receiver has no way of knowing the mode of the transmitter, there is no guarantee that a receiver will interpret messages as expected if it has been manually set to a different mode.

The recommended start up condition is Omni Mode On. This allows two instruments to be connected and played immediately without concern for selecting the instruments' basic channel. The receiver will respond to voice messages on all MIDI channels. With Omni off, a receiver would only respond to the voice messages on the Basic Channel to which it is set.

Voice Message Paths with Poly/Mono and Omni On/Off Mode Selections:



When the receiver is in Poly mode and more than one note is received on the recognized channel(s), those notes will be played simultaneously to the limit of the receiver's number of voices. The recognizable channel(s) refers to all MIDI channels when Omni is On, or to only the receiver's Basic Channel when Omni is Off.

When the receiver is in Mono mode, notes are assigned differently depending on whether Omni mode is On or Off.

MONO MODE

Mono mode is particularly useful for receiving MIDI data from guitar controllers, but can be used with keyboards and other controller devices as well. It is useful for such purposes as independent pitch bending of individual notes, portamento at specific rates between two notes, or transposition effects.

One of the reasons to use Mono mode is so that a receiver may respond in legato fashion to incoming note messages. If a Note-On is received, and then a second Note-On received without the first Note-Off being received, then the receiving instrument should change pitch to the new note, but not restart the envelopes (they should continue as if the same note was still being held). For a transmitter wishing a receiver to respond in legato fashion, the timing of the note messages would be like this:

<Note-On #1> <Note-On #2> <Note-Off #1>, etc.

MIDI rules still apply - a Note-Off must eventually be sent for every note. *Also see Legato Mode.*

OMNI-OFF/MONO

The third byte of a Mono Message specifies the number of channels in which Monophonic Voice messages are to be sent. If the letter M represents the number of acceptable MIDI channels, which may range from 1 to 16, and the letter N represents the basic channel number (1-16), then the channel(s) being used will be the current Basic Channel (N) through N+M-1 up to a maximum of 16. M=0 is a special case directing the receiver to assign all its voices, one per channel, from the Basic Channel N through 16.

TRANSMITTER: When a transmitter set to Omni-Off/Mono mode, voice messages are sent on channels N through N+M-1. This means that each individual voice (or note) is sent on a single channel. The number of transmitted channels is limited to the number of voices in the transmitter. Additional notes will be ignored. When transmitting from a 16 voice instrument whose basic channel number N is set higher than 1, N+M-1 will be greater than 16 and notes assigned to nonexistent channels above 16 should not be sent. If full 16 voice transmission is possible, the basic channel N should be set to 1. For example, a four-voice instrument set to a basic channel of 3 would transmit note messages on channels 3, 4, 5 and 6.

RECEIVER: In a receiver set to Omni-Off/Mono mode, the voice messages received in channels N through N+M-1 are assigned monophonically to its internal voices 1 through M. If N=1, and M=16 (maximum), then the messages are received on Channels 1 through 16. Should more than one Note-On message be sent for a given channel, the receivers response is not specified. Only one note (or voice) can be assigned to a given MIDI channel in this mode. M=0 is a special case directing the receiver to assign its voices, one per channel, from the basic channel N through 16, until all available voices are used.

OMNI-ON/MONO

When a transmitter is set to Omni-On/Mono mode, voice messages for a single voice are sent on channels N. If a receiver is set to Omni-On/Mono mode, then voice messages received from any voice channel will control a single voice monophonically. Regardless of the number of MIDI channels being received or the polyphony on any of them, the receiver will only play one note at a time.

TRANSMITTER: A transmitter may send a Mono message to put a receiver into Mono mode. However, since a receiver may not be capable of Mono mode, the transmitter may continue to send note messages polyphonically. Even if the transmitter and receiver are both playing monophonically, multiple Note-On messages can be sent .

RECEIVER:	When a Note-On message is sent in the Omni-On/Mono mode, the receiver will play that note regardless of channel number. If the value of M is 2 or greater when receiving a Mono On message and Omni is on, M is ignored and the receiver will still be monophonic. When Omni is on, it is inappropriate to send a Mono message with M greater than 1.

If a particular channel mode is not available on a receiver, it may ignore the message, or it may switch to an alternate mode (usually mode 1, which is Omni On/Poly).

MODES NOT IMPLEMENTED IN A RECEIVER

A transmitter could possibly request a mode not implemented in a receiver. For example, a transmitter might request Omni-Off Mono with M=2, but the receiver has only Omni-On Mono or Omni-On Poly capability. In this situation the receiver could do one of two things: (1) It can ignore the request, and no notes will sound; or (2) it can change to Omni-On Poly, and notes from both channels will play. The latter choice is recommended so that the receiver will respond to notes from the different channels.

	MONO		POLY
	M=1	M 1	
Omni Off	use Mode 2	use Mode 1	Mode 3
Omni On	Mode 2		Mode 1

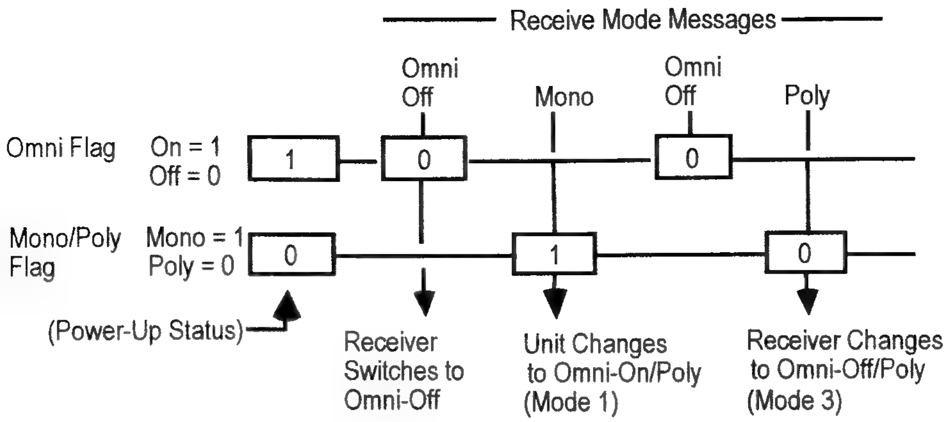
If Receiver does not have Mode 4, it can disregard messages, or act as shown here

There is no way for a transmitter to know if a receiver has responded correctly to a particular Mode message. By implementing the response outlined above, unexpected results will be minimized. If it is possible for a receiver to ignore continuous controllers, it should do so in order that pitch bend or modulation intended for a single voice will not affect all the voices in the receiver.

A transmitter may send Omni On or Omni Off messages and Poly or Mono messages in any order. A receiver should set individual flags indicating Omni On/Off and Poly/Mono.

A receiver unable to accommodate a mode message such as Omni-Off Mono may switch to an alternate mode such as Omni-On Poly. If an Omni-Off message is then received, the receiver should not change to Omni-Off unless a Poly message is also received.

It is acceptable to repeat either of the Omni On/Off or Poly/Mono messages when changing modes. For example, if a transmitter sends Omni Off Poly and later sends Omni On Poly, the retransmission of the "Poly" message should cause no problem. If a receiver cannot accommodate an Omni-Off Mono mode change from a transmitter, it should switch to an alternate mode such as Omni-On Poly as outlined above.



Receive Mode Messages

	Power Up	Omni-Off	Mono M >= 2	Omni-Off	Poly
OMNI MONO/POLY	1 0	0 0	0 1	0 1	0 0
Possible Receive Modes	1	3	N/A	N/A	3
Alternate Receive Modes			1 (Omni-On, Poly)	1 (No Change)	

OMNI-ON = 1, OMNI-OFF = 0
MONO = 1, POLY = 0

N/A = A Mode Which is not Implemented in the Receiver

ALL NOTES OFF

All Notes Off (123) is a mode message which provides an efficient method of turning off all voices turned on via MIDI. While this message is useful for some applications, there is no requirement that a receiver recognize it. Since recognition of All Notes Off is not required of the receiver, all notes should first be turned off by transmitting individual Note-Off messages prior to sending an All Notes Off.

In a MIDI keyboard instrument, notes turned on via the local keyboard should be differentiated from notes turned on via MIDI In.

In an instrument structured as shown above, it is possible that the instrument may not differentiate between MIDI In and the local keyboard commands. If an All Notes Off is received via MIDI, then *all* notes will be turned off, including those being played on the instrument's own keyboard. This is not correct implementation of the All Notes Off message. All Notes Off should only turn off those notes that were turned on via MIDI. If an instrument cannot differentiate between its local keyboard and incoming MIDI messages, All Notes Off should be ignored.

Receivers should ignore an All Notes Off message while Omni is on (Modes 1 & 2). For example, if a receiver gets an All Notes Off from a sequencer that inserts such messages whenever all keys are released on a track, and two tracks were recorded on such a sequencer (even on different MIDI channels), the All Notes Off message would cut off sustaining notes recorded on the other track.

While MIDI devices should be able to respond to the All Notes Off message, an All Notes Off message should not be sent periodically as part of normal operation. This message should only be used to indicate that the entire MIDI system is "at rest" (i.e. when a sequence has stopped). However, a receiver should respond to an All Notes Off (unless Omni is on) whenever it is received, even when the system is not "at rest".

Although other mode messages will turn off all notes, they should not be used as a substitute for the All Notes Off message when desiring to turn off all notes. When the receiver is set to Omni-Off Poly mode (Mode 3), All Notes Off will cancel Note-On messages on the basic channel only. When a receiver is set to Omni-Off Mono mode (Mode 4), All Notes Off should only cancel Note-On messages on the channel over which the message was received.

Note: See more on All Notes Off in the Additional Explanations and Application Notes.

ALL SOUND OFF

All Sound Off (120) is a mode message intended to silence all notes currently sounding by instruments receiving on a specific MIDI channel. Upon reception, all notes currently on are turned off and their volume envelopes are set to zero as soon as possible.

This message is not a replacement for the All Notes Off message, Note-Off messages, Hold Off, or Master Volume Off. The correct procedure of sending a Note-Off for each and every Note-On must still be followed.

Although originally intended for silencing notes on a MIDI sound module, the All Sound Off message may be used to turn off all lights at a MIDI-controlled lighting console or to silence and clear the audio buffer of a MIDI-controlled reverb of digital delay.

RESET ALL CONTROLLERS

When a device receives the Reset All Controllers message (121), it should reset the condition of all its controllers (continuous and switch controllers, pitch bend, and pressures) to what it considers an ideal initial state (Mod wheel to 0, Pitch Bend to center, etc.). Reception follows the same rules as All Notes Off — Ignore if Omni is On .

Sequencers that wish to implement Reset All Controllers, but want to accommodate devices that do not implement this command, should send what they believe to be the initial state of all controllers first, followed by this message. Devices that respond to this message will end up in their preferred state, while those that do not will still be in the sequencer's chosen initialized state.

LOCAL CONTROL

Channel Mode Message 122, Local Control, is used to interrupt the internal control path between the keyboard and the sound-generating circuitry of a MIDI synthesizer. If 0 (Local Off) is received the path is disconnected, keyboard data goes only to MIDI Out and the sound-generating circuitry is controlled only by incoming MIDI data. If a 7FH (Local On) is received, normal operation is restored. Local Control should be switchable from an instrument's front panel.

When a keyboard instrument is being used as a slave device via MIDI, it may be desirable to disconnect the instrument's keyboard from its internal synthesizer so that local performance cannot interfere with incoming data. This may also save scanning time and thus speed up response to MIDI information. Instruments should power-up in Local On mode. An instrument should continue to send MIDI information from its keyboard while in Local Off.

SYSTEM COMMON MESSAGES

MIDI Time Code Quarter Frame	F1H
Song Position Pointer	F2H
Song Select	F3H
Tune Request	F6H
EOX (End of Exclusive)	F7H

MTC QUARTER FRAME

For device synchronization, MIDI Time Code uses two basic types of messages, described as Quarter Frame and Full. There is also a third, optional message for encoding SMPTE user bits. The Quarter Frame message communicates the Frame, Seconds, Minutes and Hours Count in an 8-message sequence. There is also an MTC FULL FRAME message which is a MIDI System Exclusive Message.

See the separate MTC specification document for complete details.

SONG POSITION POINTER

A sequencer's Song Position (SP) is the number of MIDI beats (1 beat = 6 MIDI clocks) that have elapsed from the start of the song and is used to begin playback of a sequence from a position other than the beginning of the song. It is normally set to 0 when the START button is pressed to start sequence playback from the very beginning. It is incremented every sixth MIDI clock until STOP is pressed. If CONTINUE is pressed, it continues to increment from its current value. The current Song Position can be communicated via the Song Position Pointer message and can be changed in a receiver by an incoming Song Position Pointer message. This message should only be recognized if the receiver is set to MIDI sync (external) mode.

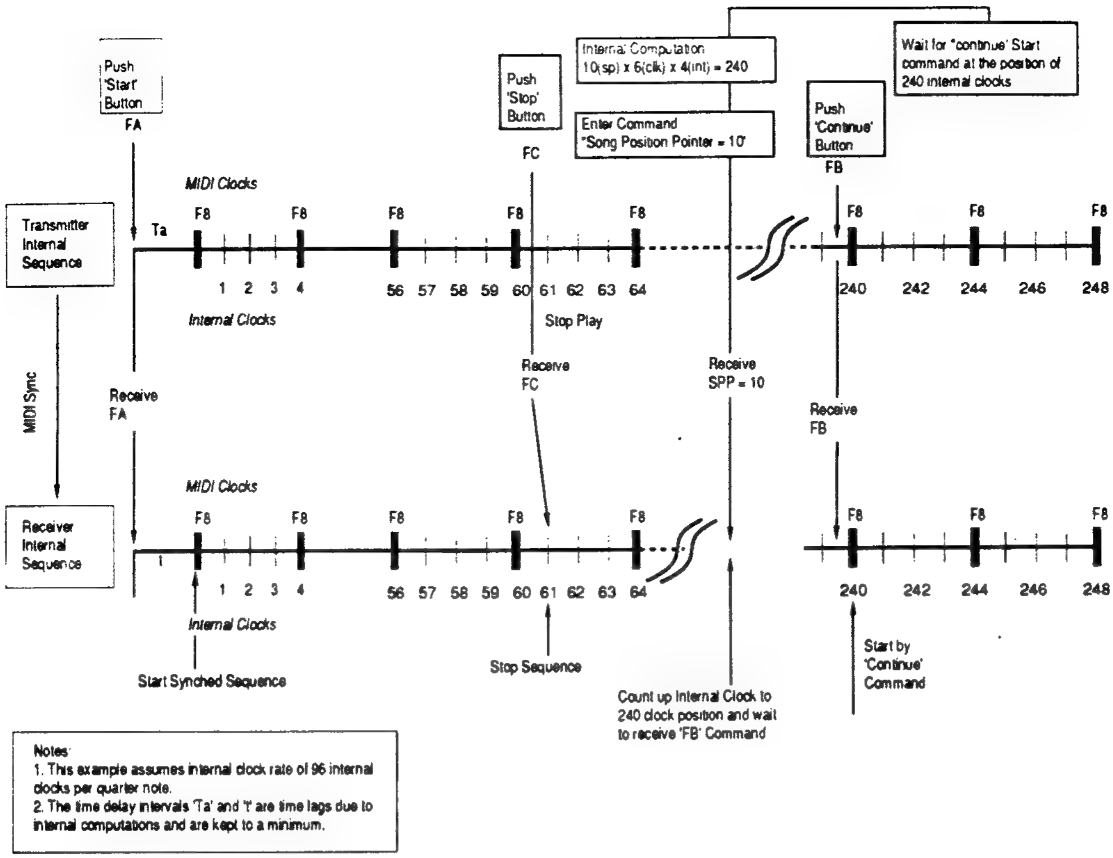
Song Position Pointer is always multiplied by 6 times the MIDI clocks (F8H). Thus the smallest Song Position change is 6 MIDI clocks, or 1/16 note. The result is then multiplied by the internal time base of the sequencer. Here is an example:

If Song Position Pointer = 10

Multiply this times 6 MIDI clocks (10 X 6 = 60)

Multiply the result (60) times the sequencer time base. If the time base is 96 clocks per beat, there are four internal clocks between each F8 so the result is 240 (60 X 4 = 240)

Set internal pointers to begin playback 240 clock tics into the sequence.



The Start message (FAH), is treated by MIDI as if it were a command comprised of a Song Position Pointer value of 0 plus a continue message (FBH).

Since the Start message and the Continue message can be received while the sequencer has been stopped by a Stop message (FCH), the sequencer should be able to start quickly in response to a Start message, even if the sequencer is in the middle of a song.

Song Position Pointer messages should be ignored if the instrument is not in MIDI sync mode (see System Real Time messages section for details on MIDI sync).

RECOMMENDED USE OF SONG POSITION POINTER

Previously it was recommended that a device wait 5 seconds after transmitting a Song Position Pointer message before it transmitted a Continue message and resumed sending MIDI Clocks. However, it is now recommended that any device receiving a Song Position Pointer (SPP) message be able to correctly receive a Continue message and subsequent MIDI Clocks while it is in the process of locating to the new position in the song. Upon locating to the new position the device must then play in sync with the device transmitting the SPP.

For example, if the transmitter sends an SPP message with a value of 4 (24 MIDI Clocks), and while locating receives a Continue as well as an additional 3 MIDI Clocks, the receiving device should begin from the 27th clock in the song.

SONG SELECT

Specifies which song or sequence is to be played upon receipt of a Start message in sequencers and drum machines capable of holding multiple songs or sequences. This message should be ignored if the receiver is not set to respond to incoming Real Time messages (MIDI Sync).

RECEPTION OF SONG POSITION AND SONG SELECT

When a device receives and recognizes a Song Position or Song Select message, it can take a relatively long time to implement the command. <u>The receiver must continue to accept MIDI clocks after a Start has been received, and increment its Song Position while it is computing and locating to the correct address in memory for playback</u>. For example, if a Song Position Pointer message is received which contains a value of 4 (24 MIDI Clocks), and during the process of locating a Continue and 3 clocks are received, the device should start playing from the point in its internal sequence corresponding to the 27th clock. If a Timing Clock message is missed while the receiver is dealing with Song Position, the receiver may not synchronize correctly. Song Position or Song Select messages may only be sent when the system is not playing.

TUNE REQUEST

Used with analog synthesizers to request that all oscillators be tuned.

EOX

Used as a flag to indicate the end of a System Exclusive transmission. A System Exclusive message starts with F0H and can continue for any number of bytes. The receiver will continue to wait for data until an EOX message (F7H) or any other non-Real Time status byte is received.

To avoid hanging a system, a transmitter should send a status byte immediately after the end of an Exclusive transmission so the receiver can return to normal operation. Although any Status Byte (except Real-Time) will end an exclusive message, an EOX should always be sent at the end of a System Exclusive message. Real time messages may be inserted between data bytes of an Exclusive message in order to maintain synchronization, and can not be used to terminate an exclusive message.

SYSTEM REAL TIME MESSAGES

Timing Clock	F8H
Start	FAH
Continue	FBH
Stop	FCH
Active Sensing	FEH
System Reset	FFH

System Real Time messages are used to synchronize clock-based MIDI equipment. These messages serve as uniform timing information and do not have channel numbers.

Real Time messages can be sent at any time and may be inserted anywhere in a MIDI data stream, including between Status and Data bytes of any other MIDI messages. Giving Real-Time messages high priority allows synchronization to be maintained while other operations are being carried out.

As most keyboard instruments do not have any use for Real-Time messages, such instruments should ignore them. It is especially important that Real-Time messages do not interrupt or affect the Running Status buffer. A Real-Time message should not be interpreted by a receiver as a new status.

TIMING CLOCK: Clock-based MIDI systems are synchronized with this message, which is sent at a rate of 24 per quarter note. If Timing Clocks (F8H) are sent during idle time they should be sent at the current tempo setting of the transmitter even while it is not playing. Receivers which are slaved to incoming Real Time messages (MIDI Sync mode) can thus phase lock their internal clocks while waiting for a Start (FAH) or Continue (FBH) command.

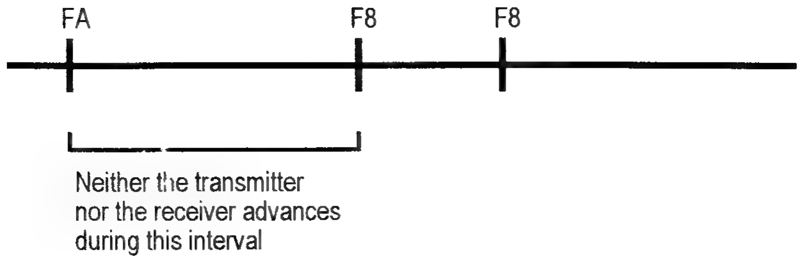
START: Start (FAH) is sent when a PLAY button on the master (sequencer or drum machine) is pressed. This message commands all receivers which are slaved to incoming Real Time messages (MIDI Sync mode) to start at the beginning of the song or sequence.

CONTINUE: Continue (FBH) is sent when a CONTINUE button is hit. A sequence will continue from its current location upon receipt of the next Timing Clock (F8H).

STOP: Stop (FCH) is sent when a STOP button is hit. Playback in a receiver should stop immediately.

START OR CONTINUE MESSAGES

When a receiver is synchronized to incoming Real Time messages (MIDI Sync mode), the receipt of a Start (FAH) or Continue (FBH) message does not start the sequence until the next Timing Clock (F8H) is received. The FA and F8 should be sent with at least 1 millisecond time between them so the receiver has time to respond. However, a receiver should be able to respond immediately to the first F8H after receiving the Start or Continue.



When the receiver is operating off of its internal clock it may ignore all Start. Stop and Continue messages or it may respond to these messages and start. stop or continue playing according to its own internal clock when these messages are received over MIDI. This decision is left up to the designer.

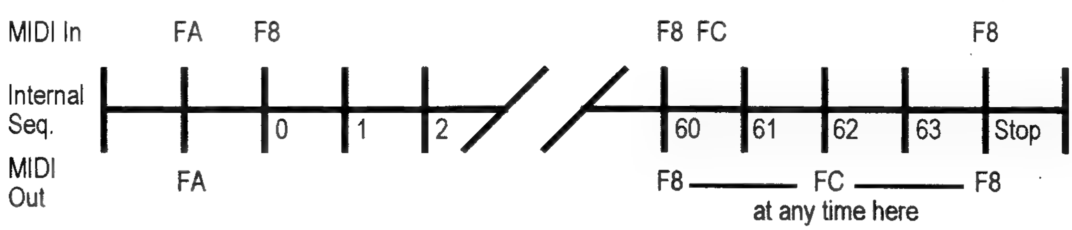
STOP MESSAGE

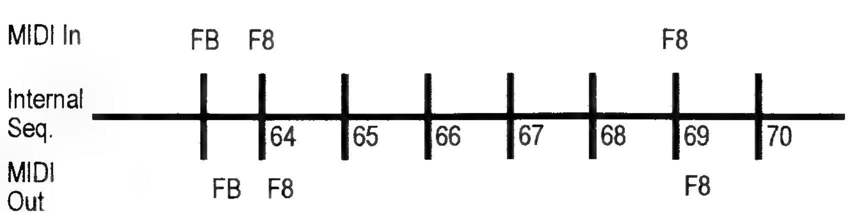
When a master sequencer is stopped it should send out the Stop message (FCH) immediately. so that any other devices slaved to it will also stop. The sequencer's internal location should be set as it was in when Stop was sent. This way. if Continue is pressed. all instruments connected to the master will continue from the same point in the song without need for a Song Position Pointer message.

Upon receiving a Stop message (FCH), a receiver should stop playing and not increment its Song Position when subsequent Timing Clock messages are received. The current Song Position should be maintained in the event that a Continue is received and the sequence is continued from the point that it was stopped. If a Song Position Pointer message is received. the device should change its internal Song Position and prepare to begin playback from the new location.

If any Note-Off messages have not been sent for corresponding Note-Ons sent before Stop was pressed. the transmitter should send the correct Note-Off messages to shut off those notes. An All Notes Off message can also be sent. but this should not be sent in lieu of the corresponding Note-Off messages as not every instrument responds to the All Notes Off message. In addition to note events. any controllers not in their initialized position (pitch wheels, sustain pedal, etc.) should be returned to their normal positions.

The following illustration shows a method to keep correct synchronization. These examples use an internal timebase of 96 pulses per 1/4 note. or 4 internal clocks per MIDI clock (F8H).

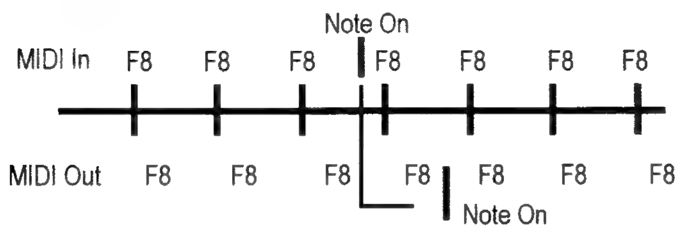




RELATIONSHIP BETWEEN CLOCKS AND COMMANDS

A sequencer may echo incoming timing and voice information out the MIDI Out port while playing its own sequenced parts. System Real Time messages should always be given time priority when the data is merged in this manner. To accomplish this, it is permissible to change the actual order of bytes to accommodate Real Time messages. However, all Real Time bytes (F8H, FAH, FBH, FCH) must be sent in the order in which they are received.

In the example below, a Note-On message is delayed slightly in order to give a priority to sending an F8H.



In order to avoid displacing clock messages in time, in addition to reversing their order with a voice message (as shown above), they may be also be inserted between the bytes of voice, common, or other messages. At no time should either an incoming clock byte or any voice message be dropped, but their order can be changed to accommodate the need for accurate timing.

A sequencer may continue sending timing clock (F8H) while it is stopped. The advantage of this is that a slaved device can know the starting tempo of a sequence just as the Start command is received.

PRIORITY OF COMMANDS

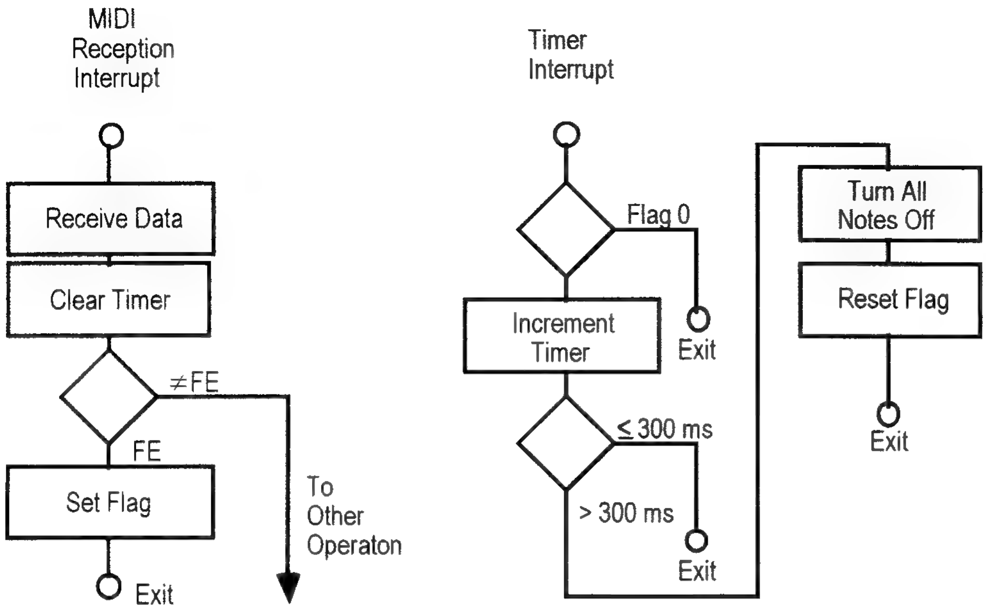
Redundant commands, such as receiving a Stop command while already stopped, or a Start or Continue command while already playing, should simply be ignored.

If a clock based device receives commands both from its front panel and via its MIDI In, priority should be given to the most recently received command. However, it is also acceptable for a device to ignore either its front panel or incoming Real Time commands depending on its current operating mode. For example, a device set to respond to incoming MIDI clocks and Real Time commands may ignore the commands received from its front panel. It may also ignore incoming Real Time commands while set to operate with its internal clock.

ACTIVE SENSING

Use of Active Sensing is optional for either receivers or transmitters. This byte (FE) is sent every 300 ms (maximum) whenever there is no other MIDI data being transmitted. If a device never receives Active Sensing it should operate normally. However, once the receiver recognizes Active Sensing (FE), it then will expect to get a message of some kind every 300 milliseconds. If no messages are received within this time period the receiver will assume the MIDI cable has been disconnected for some reason and should turn off all voices and return to normal operation. It is recommended that transmitters

transmit Active Sensing within 270ms and receivers judge at over 330ms leaving a margin of roughly 10%.
The following flowchart shows the correct method of implementing Active Sensing:



SYSTEM RESET

System Reset commands all devices in a system to return to their initialized, power-up condition. This message should be used sparingly, and should typically be sent by manual control only. It should <u>not</u> be sent automatically upon power-up and under no condition should this message be echoed.

If System Reset is recognized, the following operations should be carried out:

1) Set Omni On, Poly mode (if implemented)
2) Set Local On
3) Turn Voices Off
4) Reset all controllers
4) Set Song Position to 0
5) Stop playback
6) Clear Running Status
7) Reset the instrument to its power-up condition

SYSTEM EXCLUSIVE MESSAGES

System Exclusive F0H

System messages are not assigned to any particular MIDI channel. Thus, they will be recognized by MIDI receivers regardless of the basic channel to which they are set. System Exclusive messages, however, have a different purpose. Each instrument's System Exclusive messages (hereafter abbreviated as "Exclusive" messages) have their own special format according to an assigned manufacturer's ID number.

Exclusive messages are used to send data such as patch parameters, sampler data, or a sequencer memory bulk dump. A format which is appropriate to the particular type of transmitter and receiver is required. For example, an Exclusive message which sets the feedback level for an operator in an FM digital synthesizer will have no corresponding or meaningful function in an analog synthesizer.

Since the purpose of MIDI is to connect many kinds of musical instruments and peripheral equipment, it is best not to use Exclusive messages to convey real-time performance information (with the exception of special Universal messages described below). Performance information is best sent via Channel Voice messages in real time. Receivers should ignore non-universal Exclusive messages with ID numbers that do not correspond to their own ID.

To avoid conflicts with non-compatible Exclusive messages, a specific ID number is granted to manufacturers of MIDI instruments by the MMA or JMSC. By agreement between the MMA and JMSC when an ID number is given, the Exclusive format which is used under that ID number must be published within one year. "Published", in this context, means not only utilizing the format, but also printing the information in the product's owner's manual and/or technical materials published by the manufacturer. This is consistent with one of the fundamental purposes of MIDI, which is to publicize information and foster compatibility.

Any manufacturer of MIDI hardware or software may use the system exclusive codes of any existing product without the permission of the original manufacturer. However, they may not modify or extend it in any way that conflicts with the original specification published by the designer. Once published, an Exclusive format is treated like any other part of the instruments MIDI implementation — so long as the new instrument remains within the definitions of the published specification.

Once an Exclusive format has been published, it should not be changed with the exception of bug fixes. If a new System Exclusive format is released, it should be published in the same manner as the first version.

DISTRIBUTION OF ID NUMBERS

	American	European	Japanese	Other	Special
1 byte ID:	01 - 1F	20 - 3F	40 - 5F	60 - 7C	7D - 7F
3 byte ID:	00 00 01 00 1F 7F	00 20 00 00 3F 7F	00 40 00 00 5F 7F	00 60 00 00 7F 7F	

00 and 00 00 00 are not to be used. Special ID 7D is reserved for non-commercial use (e.g. schools, research, etc.) and is not to be used on any product released to the public. Since Non-Commercial codes would not be seen or used by an ordinary user, there is no standard format. Special IDs 7E and 7F are the Universal System Exclusive IDs..

UNIVERSAL SYSTEM EXCLUSIVE

System Exclusive ID numbers 7E (Non-Real Time) and 7F (Real Time) are Universal Exclusive IDs. used for extensions to the MIDI specification. The standardized format for both Real Time and Non-Real Time Universal Exclusive messages is as follows:

```
F0H <ID number> <device ID> <sub-ID#1> <sub-ID#2> . . . F7H
```

The `<device ID>` and `<sub-ID#1>` `<sub-ID#2>` fields are described in context below. A complete listing of the assigned Real time and Non-Real Time messages is given in TABLE VIIa.

DEVICE ID

Since System Exclusive messages are not assigned to a MIDI Channel. the Device ID (formerly referred to as the "channel" byte) is intended to indicate which device in the system is supposed to respond. The device ID 7F. sometimes referred to as the 'all call' device ID. is used to indicate that all devices should respond.

In most cases. the Device ID should refer to the physical device being addressed (the "hunk of metal and plastic" is a common term that has been used). as opposed to having the same meaning as channel or referring to a virtual device inside a physical device. For reference. this also corresponds to old USI discussions that included a "Unit ID" that was supposed to be attached to one UART and set of in/out ports.

However. there are exceptions - for example, what Device ID to use for a dual-transport tape deck and MMC commands? Some may feel more comfortable thinking of the Device ID as an "address" and allow for the possibility that a single physical unit may be powerful enough to have more than one valid address. (This also has more relevance as devices move from stand-alone units to cards in a computer.)

Therefore. Device ID is meant to refer to a single physical device or I/O port as a default. Sophisticated devices - such as multi-transport tape decks. computers with card slots. or even networks of devices - may have more than one Device ID. and such occurrences should be explained to the user clearly in the manual. From one to sixteen virtual devices may be accessed at each Device ID by use of the normal MIDI channel numbers. depending on the capabilities of the device.

SAMPLE DUMP STANDARD

A standard has been developed for sampler data dumps. It has been designed to work as an open or closed loop system. The closed loop system implements handshaking to improve speed and error recovery. This also accommodates machines that may need more time to process incoming data. The open loop system may be desired by those wishing to implement a simplified version with no handshaking.

Four of the basic messages are generic handshaking messages (ACK. NAK. Wait, & Cancel). which are also used in other applications – for example File Dump. The remaining messages are Dump Request. Dump Header. Data Packets. and a Sample Dump Extensions message. The data formats are given in hexadecimal.

GENERIC HANDSHAKING MESSAGES

ACK:

```
F0 7E <device ID> 7F pp F7
```

pp Packet number

This is the first handshaking flag. It means "Last data packet was received correctly. Start sending the next one." The packet number represents the packet being acknowledged as correct.

NAK:

```
F0 7E <device ID> 7E pp F7
```

pp Packet number

This is the second handshaking flag. It means "Last data packet was received incorrectly. Please re-send." The packet number represents the packet being rejected.

CANCEL:

```
F0 7E <device ID> 7D pp F7
```

pp Packet number

This is the third handshaking flag. It means "Abort dump." The packet number represents the packet on which the abort takes place.

WAIT:

```
F0 7E <device ID> 7C pp F7
```

pp Packet number

This is the fourth handshaking flag. It means "Do not send any more packets until told to do otherwise." This is important for systems in which the receiver (such as a computer) may need to perform other operations (such as disk access) before receiving the remainder of the dump. An ACK will continue the dump while a Cancel will abort the dump.

EOF:

```
F0 7E <device ID> 7B pp F7
```

pp packet number (ignored)

This is a new generic handshaking flag which was added for the File Dump extension, and is described fully under the File Dump heading.

DUMP HEADER

```
F0 7E <device ID> 01 ss ss ee ff ff ff gg gg gg hh hh hh ii ii ii jj F7
```

`ss ss`	Sample number (LSB first)
`ee`	Sample format (# of significant bits from 8-28)
`ff ff ff`	Sample period (1/sample rate) in nanoseconds (LSB first)
`gg gg gg`	Sample length in words (LSB first)
`hh hh hh`	Sustain loop start point word number (LSB first)
`ii ii ii`	Sustain loop end point word number (LSB first)
`jj`	Loop type (00 = forward only. 01 = backward/forward. 7F = Loop off)

DUMP REQUEST

```
F0 7E <device ID> 03 ss ss F7
```

`ss ss`	Requested sample. LSB first

Upon receiving this message. the sampler should check to see if the requested sample number falls in a legal range. If it is. the requested sample becomes the current sound number and is dumped to the requesting master following the procedure outlined below. If it is not within a legal range. the message should be ignored.

DATA PACKET

```
F0 7E <device ID> 02 kk <120 bytes> ll F7
```

`kk`	Running packet count (0-127)
`ll`	Checksum (XOR of 7E `<device ID>` 02 kk <120 bytes>)

The total size of a data packet is 127 bytes. This is to prevent MIDI input buffer overflow in machines that may want to receive an entire message before processing it. 128 bytes. or 1/2 page of memory. is considered the smallest reasonable buffer for modern MIDI instruments.

SAMPLE DUMP EXTENSIONS

All future extensions to the Sample Dump Standard will appear under the Sub-ID#1 (05) of the Universal System Exclusive Non-Real Time message.

MULTIPLE LOOP POINT MESSAGES:

These messages were added as an extension to the Sample Dump Standard, allowing for the definition of up to 16.383 pairs of loop points per sample. This cures the shortcoming of the Sample Dump Standard allowing only 1 pair of loop points to be defined per sample. It also allows modification of loop points without also having to send the sample itself.

The formats of these messages are as follows:

Loop Point Transmission (17 bytes):

`F0 7E <device ID> 05 01 ss ss bb bb cc dd dd dd ee ee ee F7`

`F0 7E <device ID>`	Universal System Exclusive Non-Real Time header
`05`	Sample Dump Extensions (sub-ID#1)
`01`	Multiple Loop messages (sub-ID#2)
`ss ss`	Sample Number (LSB first)
`bb bb`	Loop number (LSB first; 7F 7F = delete all loops)
`cc`	Loop type 00 = Forwards Only (unidirectional) 01 = Backwards/Forwards (bi-directional) 7F = Off
`dd dd dd`	Loop start address (in samples: LSB first)
`ee ee ee`	Loop end address (in samples: LSB first)
`F7`	EOX

Loop Points Request (10 bytes):

`F0 7E <device ID> 05 02 ss ss bb bb F7`

`F0 7E <device ID>`	Universal System Exclusive Non-Real Time Header
`05`	Sample Dump Extensions (sub-ID#1)
`02`	Loop Points Request (sub-ID#2)
`ss ss`	Sample Number (LSB first)
`bb bb`	Loop Number (LSB first; 7F 7F = request all loops)
`F7`	EOX

One message is sent and one loop affected per loop request or transmission, with the obvious exceptions of 'Delete All Loops' and 'Request All Loops'. If a Loop Message is sent with the same number as an existing loop, the new information replaces the old. Loop number 00 00 is the same as the sustain loop defined in the Sample Dump Standard.

SAMPLE DUMP TRANSMISSION SCENARIO

Once a dump has been requested either from the front panel or over MIDI, the dump header is sent. After sending the header, the master must time out for at least two seconds, allowing the receiver to decide if it will accept the dump (enough memory, etc.). If the master receives a Cancel, it should abort the dump immediately. If it receives an ACK, it will start sending data packets. If it receives a Wait, it will pause indefinitely until another message is received. If nothing is received within the time-out, the master will assume an open loop and begin sending packets.

A data packet consists of its own header, a packet number, 120 data bytes, a checksum, and an End Of Exclusive (EOX). The packet number starts at 00 and increments with each new packet, resetting to 00 after it reaches 7FH. This is used by the receiver to distinguish between a new data packet and one being resent. This number is followed by 120 bytes of data which form 30, 40 or 60 words (MSB first) depending on the sample format.

Each data byte consists of 7 bits. If the sample format is 8-14 bit, two bytes form a word. Sample formats of 15-21 bits require three bytes/word (yielding 40 words/packet). Sample formats of 22-28 bits require four bytes/word (yielding 30 words/packet). Information is left-justified within the 7-bit bytes and unused bits are filled in with zeros. For example, the sample word FFFH would be sent as 01111111B 01111100B. The word FFFH represent a full positive value (000H represents full negative). The checksum is the XOR of 7E <device ID> 02 <packet number> <120 bytes>.

When a sampler is receiving a data dump, it should keep a running checksum during reception. If the checksums match, it sends an ACK and wait for the next packet. If the checksums do not match, it sends a NAK and waits for the next packet. If the next packet number does not match the previous one and the sampler has no facility for receiving packets out of sequence, it should ignore the error and continue as if the checksum had matched.

When a sampler is sending a data dump, it should send a packet and watch its MIDI In port. If an ACK is received, it sends the next packet. If a NAK is received and the packet number matches that of the previous packet, it re-sends that packet. If the packet numbers do not match and the sampler has no facility to send packets out of sequence, it should ignore the NAK. If a Wait is received, the sampler should watch its MIDI IN port indefinitely for another message and process it like a normal ACK, NAK, Cancel, or illegal message (which would usually abort the dump). If nothing is received within 20ms, the sampler can assume an open loop and send the next packet.

The packet numbers are included in the handshaking flags (ACK, NAK, Cancel, Wait) in order to accommodate future machines that might have the intelligence to re-transmit specific packets out of sequence (i.e. after subsequent packets have been received).

This process continues until there are less than 121 bytes to send. The final data packet will still consist of 120 data bytes regardless of how many significant bytes actually remain. The unused bytes will be filled out with zeros. The receiver should receive and handshake on the last packet. If the receiver's memory becomes full, it should send a Cancel to the master.

DEVICE INQUIRY

The following two messages are used for device identification, and are categorized as Non-Real Time System Exclusive General Information messages (sub-ID#1 = 06).

The format of the inquiry message is as follows:

```
F0 7E <device ID> 06 01 F7
```

F0 7E <device ID>	Universal System Exclusive Non-real time header
06	General Information (sub-ID#1)
01	Identity Request (sub-ID#2)
F7	EOX

A device which receives the above message would respond as follows:

(Note that if <device ID> = 7FH then the device should respond regardless of what <device ID> it is set to.)

```
F0 7E <device ID> 06 02 mm ff ff dd dd ss ss ss ss F7
```

F0 7E <device ID>	Universal System Exclusive Non-real time header
06	General Information (sub-ID#1)
02	Identity Reply (sub-ID#2)
mm	Manufacturers System Exclusive id code
ff ff	Device family code (14 bits, LSB first)
dd dd	Device family member code (14 bits, LSB first)
ss ss ss ss	Software revision level. Format device specific
F7	EOX

Note that if the manufacturers id code (mm) begins with 00H then the above message is extended by two bytes to handle the additional manufacturers id code.

FILE DUMP

File Dump provides a protocol for transmitting files from one computer to another using MIDI. There are two primary motivations for this protocol: transmitting MIDI Files (especially tempo maps) between computers and small ROM/microcomputer-based "boxes"; and transmitting files of any type, including MIDI files, between two computers of different types. The filename is sent with the file, so that several files may be sent one after another with as little user interaction as necessary.

All File Dump messages are Exclusive Non-Real Time messages (sub-ID#1 = 07), and begin with the following header:

```
F0 7E <device ID> 07 <sub-ID#2> ss ...
```

<device ID>	device ID of message destination (7F is also acceptable here)
07	File Dump (sub-ID#1)
<sub-ID#2>	file dump message type:
01	Header
02	Data Packet
03	Request
ss	device ID of message source (7F "all-call" is NOT acceptable here)

The source device ID is included so that it may be used by the receiver of this message in all packets which it sends back to the sender of this message. In other words, if the handshake of this transfer is between device A and device B, all messages going from A to B specify B as the destination of the message, and all messages going back from B to A specify A as the destination of the message. In order to do this, the first message to B must also specify A as the source of the first message, so that B knows the device ID of who to respond to for all response messages.

REQUEST

```
F0 7E <device ID> 07 03 ss <type> <NAME> F7
```

<device ID>	device ID of request destination (will become file sender)
ss	device ID of requester (will become file receiver)
<type>	four 7-bit ASCII bytes: type of file
<NAME>	filename: 7-bit ASCII bytes terminated by the message's F7

<type> describes what type of file, in a general sense, is being requested. Only the types shown below should be used; you should only use any other type if you know that the receiver will recognize it. If the device receiving a request doesn't support the requested type, it should send the Cancel message described below.

<type>	Recommended DOS Extension	Description
MIDI	MID	It's a MIDI File
MIEX	MEX	It's a MIDIEX File
ESEQ	ESQ	It's an ESEQ File
TEXT	TXT	It's a 7-bit ASCII Text File
BIN<space>	BIN	It's a binary file (such as any MS-DOS file)
MAC<space>	MAC	It's a Macintosh file (with MacBinary header)

If <type> is MAC, this means a Macintosh file is being requested. Because Macintosh files contain two "forks," and other important Finder information, they are sent as their MacBinary image. Note that programs wishing to transmit only MIDI Files, even on the Macintosh, won't need to worry about MacBinary, because MIDI Files must always use the MIDI designation to be universally recognized as MIDI Files.

The filename may be any length, and may be omitted entirely. If it is omitted, it means "whatever is currently loaded." The filename may contain only printable ASCII characters (20H through 7EH). Colons and backslashes may optionally be interpreted as path specifiers: these characters should be avoided in filenames if this behavior is not desired by the user. If the device receiving the request message does not have a file system, it should send whatever is currently loaded, using a null filename.

If the device receiving the request message is a computer, it should initiate a transfer if it recognizes the filename, or if there is no filename but there is a "currently loaded" file. If these conditions aren't met, it may either prompt the user for a valid filename (displaying the filename supplied in the dump message), or just send the Cancel message back to the requester. If the user is to be prompted, the Wait message should be sent to the requester so that it knows that it may be awhile before the transfer is initiated or a Cancel is to be sent.

HEADER

```
F0 7E <device ID> 07 01 ss <type> <length> <NAME> F7
```

`<device ID>`	device ID of requester/receiver
`ss`	device ID of sender
`<type>`	four 7-bit ASCII bytes: type of file
`<length>`	four 7-bit bytes: actual (un-encoded) file length, LSB first
`<NAME>`	filename: 7-bit ASCII bytes terminated by the message's F7

<type> and <NAME> are exactly as described in the Dump Request message.

If the length of the file is not known (because it will be converted on the fly), zero may be sent as the length.

If the sender is a small ROM-based "box" without files, it need not send a filename. If it is a computer, and there is a filename associated with it, it should be sent in the header. As described above, it may be any length, must only contain printable ASCII characters, and may contain path description characters. For maximum compatibility, no path information should be sent. DOS-like machines should send the file extension as part of the name, separated by a period, with no trailing spaces before the period.

If the receiver is a computer, and if the program running supports receiving files, it should modify the filename if necessary to make it appropriate for its file system. For instance, if it is a DOS machine, and the given filename contains a period, it should interpret everything after the period as the file's extension. If there is no period, it should use the appropriate extension listed above. If it is running interactively, it should prompt the user with the filename supplied in the dump message, so that the user can modify it if desired, if no name is sent, or if a file by that name already exists. If the user is to be prompted, the Wait message should be sent to the sender so that it knows that it may be awhile before the transfer is continued or a Cancel is sent.

If the receiver is a small ROM-based "box" without files, or a program on a computer which only expects this protocol to replace the file currently in memory, it should simply ignore the filename and replace its current memory contents with the contents of the transmitted file, if the file is a supported type.

DATA PACKET

```
F0 7E <device ID> 07 02 <packet #> <byte count> <data> <chksm> F7
```

<device ID>	device ID of receiver
<packet #>	one-byte packet count
<byte count>	one-byte packet size: number of encoded data bytes minus one
<data>	the data, encoded as described below
<chksm>	one-byte checksum (XOR of all bytes which follow F0 up to the checksum byte, similar to sample dump)

The total size of a data packet may be slightly larger than for sample dump: 137 bytes maximum. The packet number starts at 00 and increments with each new packet, resetting to 00 after it reaches 7FH. This is used by the receiver to detect missed packets. The byte count is the number of encoded data bytes minus one: for example, 64 stored bytes of the file are encoded in 74 transmitted bytes (as described below): the byte count would be 73. (Subtracting one allows sending 128 transmitted data bytes: one would never need to send zero bytes).

Instead of nibblizing, which would double transmission time, the data is "7-bit-ized" so that the transmission time is more like 12% more than sending it as 8-bit (which isn't possible over MIDI). Each group of seven stored bytes is transmitted as eight bytes. First, the sign bits of the seven bytes are sent, followed by the low-order 7 bits of each byte. (The reasoning is that this would make the auxiliary bytes appear in every 8th byte without exception, which would therefore be slightly easier for the receiver to decode.) The seven bytes:

```
AAAAaaaa BBBBbbbb CCCCcccc DDDDdddd EEEEeeee FFFFffff GGGGgggg
```

are sent as:

```
0ABCDEFG
0AAAaaaa 0BBBbbbb 0CCCcccc 0DDDdddd 0EEEeeee 0FFFffff 0GGGgggg
```

From a buffer to be encoded, complete groups of seven bytes are encoded into groups of eight bytes. If the buffer size is not a multiple of seven, there will be some number of bytes left over after the groups of seven are encoded. This short group is transmitted similarly, with the sign bits occupying the most significant bits of the first transmitted byte. For example:

```
AAAAaaaa BBBBbbbb CCCCcccc
```

are transmitted as:

```
0ABC0000 0AAAaaaa 0BBBbbbb 0CCCcccc
```

Since the maximum packet size is 128 transmitted bytes, this corresponds to sixteen groups of seven bytes, or 112 stored bytes.

HANDSHAKING FLAGS

For handshaking messages, the same generic set originally created for Sample Dump Standard – plus a new EOF message – are to be used (Non-Real Time sub-ID#1 = 7B-7F). Since these first four message were explained in the Sample Dump section, only newly significant information will be presented here.

```
F0 7E <device ID> <sub-ID#1> pp F7
```

<device ID>	device ID of packet sender (message destination)	
<sub-ID#1>	handshake message:	
7B		End of File
7C		Wait
7D		Cancel
7E		NAK
7F		ACK
pp	packet number	

NAK:

```
F0 7E <device ID> 7E pp F7
```

<device ID>	device ID of packet sender (ACK receiver)
pp	packet number

This should be sent whenever the length of a message was wrong, or the checksum was incorrect. After receiving a NAK, the sender should resend the packet. After sending a NAK, the receiver should expect the same packet to be resent. If the same packet has an error three consecutive times, a Cancel should be sent instead of a NAK. If the packet number was wrong, such as if a packet (or a NAK) was missed, the Cancel message should be sent instead of a NAK.

ACK:

```
F0 7E <device ID> 7F pp F7
```

<device ID>	device ID of packet sender (ACK receiver)
pp	packet number

The packet number represents the packet being acknowledged as correct. The packet number in the ACK responding to the Header is undefined.

WAIT:

```
F0 7E <device ID> 7C pp F7
```

<device ID>	device ID of Wait receiver
pp	packet number (ignored)

This handshaking flag is used after receiving a File Header, Data Packet, or File Dump Request. When responding to a Header it means "Do not send data packets until you receive an ACK (or a Cancel)." When responding to a data packet, it means "Do not send any more packets until you

receive an ACK or NAK (or a Cancel)." When used in response to a File Dump Request, it means "Your File Header (or a Cancel) will follow soon – be patient."

This message is important for systems in which the receiver may need to perform other operations, such as disk access or prompting the user, before processing the remainder of the dump. A slow device may in fact wish to transmit a Wait every time it receives a File Header, Data Packet, or File Dump Request, thus giving itself unlimited time in which to digest the received data and respond appropriately.

CANCEL:

```
F0 7E <device ID> 7D pp F7
```

`<device ID>`	device ID of Cancel receiver
`pp`	packet number (ignored)

This handshaking flag may be used at any time. It means "Abort dump." The packet number represents the packet on which the abort takes place, but is ignored by the receiver. This may be sent by either the sender or receiver when any error is detected, such as incorrect packet numbers in a data packet or a handshake message: or when a dump is canceled by the user. If the sender aborts a transmission, it should use the receiver's device ID in the Cancel message (which it put in the header (`<device ID>`) in the first place). If the receiver aborts a transmission, it should use the sender's device ID in the Cancel message (which the sender put in the header (ss)).ACK

END OF FILE (EOF):

```
F0 7E <device ID> 7B pp F7
```

`<device ID>`	device ID of receiver
`pp`	packet number (ignored)

This is the fifth generic handshaking flag within MIDI, sub-ID#1 (7B). After sending the last packet of a lengthy message (such as File Dump), the sender must send an EOF message to inform the receiver that the entire file has been sent. This is critical if the length in the File Dump Header is 0 (which means that the file length is unknown), because this is the only way the receiver can know the transmission is complete and correct. This message must be sent even if the correct length is known at the beginning. EOF requires no response from the receiver.

FILE DUMP TRANSMISSION SCENARIO

The File Dump Request is optional. A device may request a file (or memory contents), using the Request message, or a user may initiate a file dump without a request message being sent. Within 200 msec after receiving the Request message EOX (F7), the sender must respond with a File Dump header, Wait, or Cancel. If it responds with Wait, it may send a File Dump header or a Cancel message whenever it's good and ready.

The sender then sends a File Dump header message. Within 200 msec after receiving the Header EOX (F7), the receiver must respond with ACK, Wait, or Cancel. If it responds with Wait, it may send an ACK or a Cancel message whenever it's good and ready. If the sender does not receive any message during this time, it assumes open loop transmission, and proceeds as if an ACK had been received.

The sender then sends a data packet. As the receiver receives the data packet, it keeps a running checksum. If the checksums match, and it can deal with the data immediately, it sends an ACK and

waits for the next packet. If it needs more than 50 msec to store the data, it sends a Wait message. (After storing the data, it then sends an ACK to continue the process). If the checksums do not match, or if the length is wrong, it sends a NAK and waits for the same packet to be resent. If the packet number is not the one it was expecting, it sends a Cancel message and ignores all further data packets until a new header is sent (in the open-loop case, the sender won't ever receive a Cancel message). If the receiver's memory ever becomes full, even during the last packet, it should send a Cancel to the sender.

When a device is sending a data dump, it should send a packet and watch its MIDI IN port. If an ACK is received, it should send the next packet immediately. If a NAK is received and the packet number matches that of the previous packet, it re-sends that packet. If the packet number of an ACK or a NAK do not match the number of the packet just sent, the sender should send a Cancel message, and abort the transmission. If a Wait is received, it should watch its MIDI IN port indefinitely for another message. If it receives an ACK or NAK, it should process it normally, and continue; if it receives a Cancel or an illegal message, it should abort the dump process. If nothing is received in 50 msec after a data packet or 200 msec after a header, it can assume an open loop and send the next packet.

After the receiver ACKs the last packet, the sender transmits an EOF. No ACK is required for this message. The file dump is then complete.

Any packet may contain any number of bytes, up to 128 encoded data bytes. Most devices probably will transmit several packets of equal size, and send what's left over as a final packet. However, the receiver should never make any assumption about packet size.

MIDI TUNING

This is an addition to the MIDI specification which allows the sharing of "microtunings" (user-defined scales other than 12-tone equal temperament) among instruments of different manufacture, and the switching of these tunings during real-time performance.

The messages include:

- bulk tuning dump request (non-real-time)
- bulk tuning dump (non-real-time)
- single-note tuning change (real-time)

Even though the first two messages are in the Universal Non-Real Time area and the last in the Real Time area, they keep the same sub-IDs to more obviously group them and possibly ease the parsing of them. Single Note Retuning is a part of the proposal which allows retuning of individual MIDI note numbers to new frequencies in real time as a performance control.

The standard does not attempt to dictate how a manufacturer implements microtuning, but provides a general means of sharing tuning data among different instruments.

This goal does require shared assumptions which have some architectural implications. The standard requires that any of the 128 defined MIDI key numbers (or at least those MIDI key numbers covered by the instrument's playable range) be tunable to any frequency within the proposed frequency range. The standard also strongly suggests, but does not enforce, an exponential (constant cents) rather than linear (constant Hertz) tuning resolution across the instrument's frequency range.

The standard permits the changing of tunings in real-time, both by the selection of presets and on a per-note basis. When a sounding note is affected by either real-time tuning message, the note should instantly be re-tuned to the new frequency while it continues to sound; this change should occur without glitching, forced Note-Offs, re-triggering or other audible artifacts (see section 4, "Additional").

The standard provides for 128 tuning memory locations (programs). As with the MIDI program change message, this is a maximum value. An instrument supporting the standard may have any lesser number of tuning programs. The standard requires only that all existing tuning programs respond to the messages as specified (See section 3, "Continuous Controller Messages").

Although directly applicable to some existing instruments, the standard attempts to define a coherent framework within which the designers of future instruments can profitably work. It is hoped that by providing this framework the standard will make microtunability more easily implemented and more common on MIDI instruments.

FREQUENCY DATA FORMAT

The frequency resolution of the standard should be stringent enough to satisfy most demands of music and experimentation. The standard provides resolution somewhat finer than one-hundredth of a cent. Instruments may support the standard without necessarily providing this resolution in their hardware: the standard simply permits the transfer of tuning data at any resolution up to this limit.

Frequency data shall be sent via system exclusive messages. Because system exclusive data bytes have their high bit set low, containing 7 bits of data, a 3-byte (21-bit) frequency data word is used for specifying a frequency with the suggested resolution. An instrument which does not support the full suggested resolution may discard any unneeded lower bits on reception, but it is preferred where possible that full resolution be stored internally, for possible transmission to other instruments which can use the increased resolution.

Frequency data shall be defined in units which are fractions of a semitone. The frequency range starts at MIDI note 0, C = 8.1758 Hz, and extends above MIDI note 127, G = 12543.875 Hz. The first byte of the frequency data word specifies the nearest equal-tempered semitone below the frequency. The next two bytes (14 bits) specify the fraction of 100 cents above the semitone at which the frequency lies. Effective resolution = 100 cents / 2^{14} = .0061 cents.

One of these values (7F 7F 7F) is reserved to indicate not frequency data but a "no change" condition. When an instrument receives these bytes as frequency data, it should make no change to its stored frequency data for that MIDI key number. This is to prevent instruments which do not use the full range of 128 MIDI key numbers from sending erroneous tuning data to instrument which do use the full range. The three-byte frequency representation may be interpreted as follows:

```
0xxxxxxx  0abcdefg  0hijklmn
```

`xxxxxxx`	= semitone
`abcdefghijklmn`	= fraction of semitone, in .0061-cent units

Examples of frequency data:

`00 00 00 =`	8.1758 Hz	(C – normal tuning of MIDI key no. 0)
`00 00 01 =`	8.2104 Hz	
`01 00 00 =`	8.6620 Hz	
`0C 00 00 =`	16.3516 Hz	
`3C 00 00 =`	261.6256 Hz	(middle C)
`3D 00 00 =`	277.1827 Hz	(C# – normal tuning of MIDI key no. 61)
`42 7F 7F =`	439.9984 Hz	
`43 00 00 =`	440.0000 Hz	(A-440)
`43 00 01 =`	440.0016 Hz	
`78 00 00 =`	8372.0190 Hz	(C – normal tuning of MIDI key no. 120)
`78 00 01 =`	8372.0630 Hz	
`7F 00 00 =`	12543.8800 Hz	(G – normal tuning of MIDI key no. 127)
`7F 00 01 =`	12543.9200 Hz	
`7F 7F 7E =`	13289.7300 Hz	(top of range)
`7F 7F 7F =`	no change	(reserved)

BULK TUNING DUMP REQUEST

A bulk tuning dump request is as follows:

```
F0 7E <device ID> 08 00 tt F7
```

`F0 7E`	Universal Non-Real Time SysEx header
`<device ID>`	ID of target device
`08`	sub-ID#1 = MIDI Tuning Standard
`00`	sub-ID#2 = 00H, bulk dump request)
`tt`	tuning program number (0 – 127)
`F7`	EOX

The receiving instrument shall respond by sending the bulk tuning dump message described in the following section for the tuning number addressed.

BULK TUNING DUMP

A bulk tuning dump comprises frequency data in the 3-byte format outlined in section 1. for all 128 MIDI key numbers. in order from note 0 (earliest sent) to note 127 (latest sent). enclosed by a system exclusive header and tail. This message is sent by the receiving instrument in response to a tuning dump request.

```
F0 7E <device ID> 08 01 tt <tuning name> [xx yy zz] ... chksum F7
```

F0 7E	Universal Non-Real Time SysEx header
<device ID>	ID of responding device
08	sub-ID#1 = MIDI Tuning Standard
01	sub-ID#2 = 01H. bulk dump reply)
tt	tuning program number (0 – 127)
<tuning name>	16 ASCII characters
[xx yy zz]	frequency data for one note (repeated 128 times)
chksum	checksum (XOR of 7E <device ID> nn tt <388 bytes>)
F7	EOX

If an instrument does not use the full range of 128 MIDI key numbers. it may ignore data associated with un-playable notes on reception. but it is preferred where possible that the full 128-key tuning be stored internally. for possible transmission to other instruments which can use the increased resolution. On transmission. it may if necessary pad frequency data associated with un-playable notes with the "no change" frequency data word defined above. For keys in the instrument's key range. the pitch that is sent should be the pitch that key would play if it were received as part of a note-on message. For keys outside the key range. 7F 7F 7F may be sent.

SINGLE NOTE TUNING CHANGE (REAL-TIME)

The single note tuning change message (Exclusive Real Time sub-ID#1 = 08) permits on-the-fly adjustments to any tuning stored in the instrument's memory. These changes should take effect immediately. and should occur without audible artifacts if any affected notes are sounding when the message is received.

```
F0 7F <device ID> 08 02 tt ll [kk xx yy zz] F7
```

F0 7F	Universal Real Time SysEx header
<device ID>	ID of target device
08	sub-ID#1 (MIDI Tuning Standard)
02	sub-ID#2 (02H, note change)
tt	tuning program number (0 – 127)
ll	number of changes (1 change = 1 set of [kk xx yy zz])
[kk	MIDI key number
xx yy zz]	frequency data for that key (repeated 'll' number of times)
F7	EOX

This message also permits (but does not require) multiple changes to be embedded in one message. for the purpose of maximizing bandwidth. The number of changes following is indicated by the byte ll; the total length of the message equals 8 + (ll x 4) bytes.

If an instrument does not support the full range of 128 MIDI key numbers, it should ignore data associated with un-playable notes on reception.

This message can be used to make changes in inactive (background) tunings as well. This message may also, at the discretion of the manufacturer, be transmitted by the instrument under particular circumstances (for example, while holding down one or more keys and pressing a "send-single-note-tuning" front panel button).

CHANGING TUNING PROGRAMS

A registered parameter number shall be allotted to select any of the instrument's stored tunings as the "current" or active tuning. Instruments which permit the storage of multiple microtunings should respond to this message by instantly changing the "current" tuning to the specified stored tuning. This change takes effect immediately and must occur without audible artifacts (notes-off, resets, re-triggers, glitches, etc.) if any affected notes are sounding when the message is received.

As with the MIDI program change message, no assumptions are made as to the underlying architecture of the instrument. For instance, in cases where layered or multi-timbral sounds might be assigned to different tunings, so that more than one tuning might be active, the manufacturer may decide how best to interpret this message. The basic channel number might prove useful in discriminating between multiple active tunings, or a certain range of tuning programs might be set aside and defined as active.

The message is sent as a registered parameter number controller message, followed by either a data entry, data increment, or data decrement controller message, e.g. (with running status shown):

```
Bn 64 03 65 00 06 tt      (data entry)
Bn 64 03 65 00 60 7F      (data increment)
Bn 64 03 65 00 61 7F      (data decrement)
       n        = basic channel number
       tt       = Tuning Program number (1-128)
```

Likewise, a Tuning Bank Change Registered Parameter number is also assigned as follows:

```
Bn 64 04 65 00 06 tt      (data entry)
Bn 64 04 65 00 60 7F      (data increment)
Bn 64 04 65 00 61 7F      (data decrement)
       n        = basic channel number
       tt       = Tuning Bank number (1-128)
```

For maximum flexibility, this Bank Number is kept separate from the normal Program Change Bank Select (controller #00). However, an instrument may wish to link the two as a feature for the user, especially if a tuning bank is stored alongside a patch parameter bank (for example, on a RAM cartridge).

If an instrument receives a Tuning Program or Bank number for which it has no Program or Bank, it should ignore that message. Standard mappings of "common" tunings to program numbers are not being proposed at this time.

<u>Additional</u>

There is some question as to whether instantaneous response to real-time tuning changes is desirable in every circumstance. In some performance situations it makes more sense if a tuning change affect only

those notes which occur subsequent to the change, and not affect sounding notes. But there are also situations in which tuning changes should take place instantaneously, as specified in the standard, and should affect sounding notes without disrupting their continuity.

If the instrument responds well in the latter situation, some work-around is possible for the former. The reverse is not true. Therefore the standard requires that tuning changes immediately affect sounding notes. Manufacturers might, however, consider implementing a switchable "instantaneous/next-note-on" option within an instrument.

Single Note Retuning is intended for performance. Because of there are two primary concerns: 1) the RAM required for temporary copies of tuning tables; and 2) the computational load of smoothly updating the pitch of affected active notes. It is clear that in order to recognize the Single Note Retune message, a copy of the current Tuning Program needs to be kept in RAM. In a multi-timbral environment there is potentially a copy for each virtual instrument. A high-end instrument could afford the upwards of 8K of RAM needed for per-virtual-instrument copies. More modest instruments may choose to only implement one alterable RAM table and either make it available only to the basic channel virtual instrument or require that all instruments share the same tuning. Provided that it is explained in the user's manual, any of these methods is acceptable.

Additional information on alternate tunings:
The Just Intonation Network
MIDI Tuning Standards Committee
535 Stevenson St.
San Francisco, CA 94103

GENERAL MIDI SYSTEM MESSAGES

There is a defined set of Universal Non-Real Time SysEx messages for General MIDI (sub-ID#1 = `09`). The current messages (below) turn GM mode on/off on a sound module (should it have more than one mode of operation):

Turn General MIDI System On:

```
F0 7E <device ID> 09 01 F7
```

`F0 7E`	Universal Non-Real Time SysEx header
`<device ID>`	ID of target device (suggest using `7F` 'All Call')
`09`	sub-ID#1 = General MIDI message
`01`	sub-ID#2 = General MIDI On
`F7`	EOX

Turn General MIDI System Off:

```
F0 7E <device ID> 09 02 F7
```

`F0 7E`	Universal Non-Real Time SysEx header
`<device ID>`	ID of target device (suggest using `7F` 'All Call')
`09`	sub-ID#1 = General MIDI message
`02`	sub-ID#2 = General MIDI Off
`F7`	EOX

MTC FULL MESSAGE, USER BITS, REAL TIME CUEING

While MTC Quarter Frame messages (System Common) handle the basic running work of the system. they are not suitable for use when equipment needs to be fast-forwarded or rewound. located or cued to a specific time. as sending them continuously at accelerated speeds would unnecessarily clog up or outrun the MIDI data lines. For these cases. MTC Full Messages are used. which encode the complete time into a single message. After sending a Full Message. the time code generator can pause for any mechanical devices to shuttle (or "autolocate") to that point. and then resume running by sending quarter frame messages.

Universal System Exclusive Real Time sub-ID#1 (01) is used for the MTC Full Message. and for defining MTC User Bits. Real Time sub-ID#1 (05) is used for MIDI Cueing.

See the separate MTC Detailed Specification for complete details.

MIDI SHOW CONTROL

The purpose of MIDI Show Control is to allow MIDI systems to communicate with and to control dedicated intelligent control equipment in theatrical. live performance. multi-media. audio-visual and similar environments. Applications may range from a simple interface through which a single lighting controller can be instructed to GO. STOP or RESUME. to complex communications with large. timed and synchronized systems utilizing many controllers of all types of performance technology.

MIDI Show Control uses a single Universal System Exclusive Real Time sub-ID#1 (02) for all Show commands (transmissions from Controller to Controlled Device).

See the separate MSC Detailed Specification for complete details.

NOTATION INFORMATION

Universal System Exclusive Real Time subID#1 (03) is used for communicating musical structure information in real time.

The messages include Bar Marker. Time Signature (Delayed). and Time Signature (Immediate).

BAR MARKER

The Bar Marker message specifies that the next MIDI clock received is the first clock of a measure. and thus a new bar.

The message format is as follows:

```
F0 7F <device ID> 03 01 aa aa F7
```

`F0 7F`	Universal Real Time SysEx Header		
`<device ID>`	ID of target device (default = 7F [all])		
`03`	sub ID#1 = Notation Information		
`01`	sub ID#2 = Bar Number Message		
`aa aa`	bar number: lsb first		
	`[00 40]`		not running
	`[01 40] - [00 00]`		count-in
	`[01 00] - [7E 3F]`		bar number in song
	`[7F 3F]`		running: bar number unknown
`F7`	EOX		

The numbering system uses the largest possible negative number as the "not running" flag: count-in bars are negative numbers until they reach zero, which is the last bar of count-in (systems that have only 1 bar of count-in don't have to deal with negative numbers – just count from "zero" on up): bar numbers then increment through positive numbers, with the highest positive number reserved as "running, but I don't know the bar number" (or the bar number has exceeded 8K).

If MIDI clocks (F8s) are also being sent. this bar number takes effect at the next received F8. If MTC but no MIDI clocks are being sent. this bar number takes effect at the next received F1 xx. It may be displayed as soon as received (in the event that it was sent while a drum machine or sequencer is paused, but has located to a new section of the song).

Please note that this message is intended for information and high-level synchronization as opposed to low-level synchronization. and should not be taken as a substitute for other MIDI timing messages.

The Bar Marker message is critical for other Notation messages (such as Time Signature) which have the option of taking effect immediately or on the next received Bar Marker message. In the later case. extra information can be sent at any time during the previous bar without taking effect. This will minimize clogging by allowing enough room between the last F8/F1 xx of a bar and the first F8/F1 xx of the next. With the Bar Marker being sent every bar, a receiver does not have to keep track of MIDI clocks to know exactly where it is.

Therefore. it is strongly suggested that the Bar Number be sent immediately after the last F8 or F1 xx message of the previous bar. to prevent possible clogging. jitter, and/or message transposition (MIDI mergers may also want to be sensitive to this message to prevent it getting delayed past a following F8).

TIME SIGNATURE

The Signature Messages are used to communicate a new time signature to a receiving device. There are two forms. Immediate and Delayed. The Immediate form (sub id #2 = 02H [bit 6 = reset]) takes effect upon receipt (or on the next received MIDI clock if slaved to MIDI sync). The Delayed form (sub-ID#2 = 42H [bit 6 = set]) takes effect upon the receipt of the next Bar Marker message. However, it may be displayed immediately.

Time Signature (Immediate):

```
F0 7F <device id> 03 02 ln nn dd bb cc bb [nn dd...]   F7
```

F0 7F	Universal Real Time SysEx header
<device id>	ID of target device (default = 7F [all])
03	sub-ID#1 = Notation Information
02	sub-ID#2 = Time Signature - Immediate
ln	number of data bytes to follow
nn	number of beats (numerator) of time signature
dd	beat value (denominator) of time signature (negative power of 2)
cc	number of MIDI clocks in a metronome click
bb	number of notated 32nd notes in a MIDI quarter note
[nn dd...]	additional pairs of time signatures to define a compound time signature within the same bar.
F7	EOX

Time Signature (Delayed):

```
F0 7F <device id> 03 42 ln nn dd bb cc bb [nn dd...]   F7
```

F0 7F	Universal Real Time SysEx header
<device id>	ID of target device (default = 7F [all])
03	sub-ID#1 = Notation Information
42	sub-ID#2 = Time Signature - Delayed
ln	number of data bytes to follow
nn	number of beats (numerator) of time signature
dd	beat value (denominator) of time signature (negative power of 2)
cc	number of MIDI clocks in a metronome click
bb	number of notated 32nd notes in a MIDI quarter note
[nn dd...]	additional pairs of time signatures to define a compound time signature within the same bar.
F7	EOX

The additional data in [nn dd...] must always be in pairs. If there are not additional time signatures specified, ln (the length of the data) = 4. It is incremented by multiples of 2 for every extra time signature pair that exists within the bar.

The data format here duplicates that of the Standard MIDI File Time Signature Meta Event (FF 58), with extra bytes for compound time signatures. The bytes for the compound time signatures were added at the end so that the current Meta Event could be extended to match the format of this message, while keeping the leading bytes of the event the same.

The burden is placed on the transmitter to indicate ahead of time what the time signature will be in the next bar. It is not the responsibility of the receiver to count clocks and decode it. It is intended that interpretation of the Notation family of messages be made as simple as possible for the receiver so that devices with displays (which may not be following MIDI clocks) could easily pass useful information to the user.

DEVICE CONTROL

MASTER VOLUME AND MASTER BALANCE

These messages are intended to produce the same effect as volume and balance controls on a stereo amplifier. They are intended mainly for General MIDI instruments (so that one Master Volume control can simultaneously fade out all the layers in a sound module, for example), although there may be wider applications.

Because these messages are intended to address "devices" as opposed to MIDI "channels" they have been defined as Universal Real Time System Exclusive messages (sub-ID#1 = `04`). The corresponding "channel" messages are the controllers Channel Volume (formerly Main Volume) (CC number 07) and Balance (CC number 08).

Master Volume:

```
F0 7F <device id> 04 01 vv vv F7
```

`F0 7F <device id>`	Universal Real Time SysEx header
`04`	sub-ID#1 = Device Control
`01`	sub-ID#2 = Master Volume
`vv vv`	Volume (lsb first): 00 00 = volume off
`F7`	EOX

Master Balance:

```
F0 7F <device id> 04 02 bb bb F7
```

`F0 7F <device id>`	Universal Real Time SysEx header
`04`	sub-ID#1 = Device Control
`02`	sub-ID#2 = Master Balance
`bb bb`	Balance (lsb first): 00 00 = hard left; 7F 7F = hard right
`F7`	EOX

In order to properly respond to these messages and their channel-aimed counterparts, a device must internally track three volume and two balance scalars as follows:

1. Received on its own ID (which matches its knob on the front panel: if no knob or if knob is not scanned then power up default is set at full volume)

2. Received on the 'All Call' or 'broadcast' ID (`7F`)

3. Channel messages.

This way, each virtual/channel-based instrument can be individually mixed, then a device could be individually scaled, and then all devices could be brought down together without forgetting their individual levels.

MIDI MACHINE CONTROL

MIDI Machine Control is a general purpose protocol which initially allows MIDI systems to communicate with and to control some of the more traditional audio recording and production systems. Applications may range from a simple interface through which a single tape recorder can be instructed to PLAY. STOP. FAST FORWARD or REWIND. to complex communications with large. time code based and synchronized systems of audio and video recorders. digital recording systems and sequencers.

MIDI Machine Control uses two Universal Real Time System Exclusive messages (sub-ID#1's). one for Commands (transmissions from Controller to Controlled Device). and one for Responses (transmissions from Controlled Device to Controller). (sub-ID#1 = 06, 07)

See the separate MMC Detailed Specification for complete details.

ADDITIONAL EXPLANATIONS AND APPLICATION NOTES

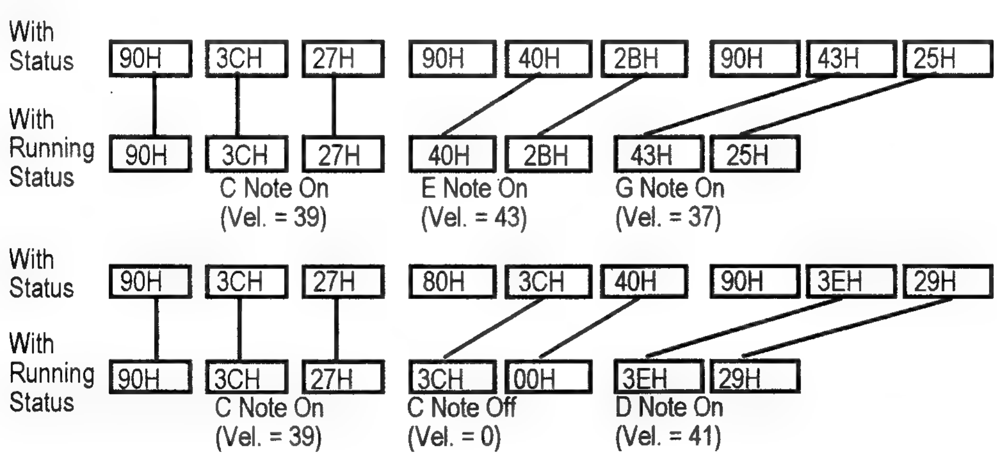
RUNNING STATUS

Running status is a convenient short cut in transmission of data which saves time and makes it easier to minimize delays of transmitted MIDI data from the actual performance. With Running Status, after the first message, additional messages of the same type (i.e. Note On messages on the same MIDI channel) are sent without repeating the status byte for every message. Receivers must understand that if a data byte is received as the first byte of a message, the most recent, or "running" status is assumed.

For example, a note is normally played by transmitting a Note On Status Byte (90H) followed by the key number value (0kkkkkkk) and the velocity value bytes (0vvvvvvv). With Running Status, all additional notes on the same MIDI channel can be played by simply transmitting the key number and velocity bytes. As long as all following data consists of Note Ons on the same MIDI channel the Note On status byte need not be sent again.

Running Status is most useful for Note On and continuous controller messages. As notes can be turned off by sending a Note On with a velocity value of 0, long strings of note messages can be sent without sending a Status byte for each message. If the Note Off (8nH) message is used to turn notes off, a status byte must be sent.

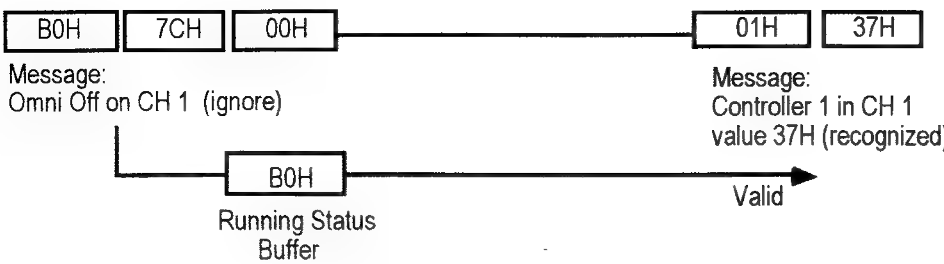
The following is an example of Running Status. On the top is a complete data stream with one Status Byte for each pair of Data Bytes. Below that is a compressed data stream with only one Status Byte:



While the above examples pertain to Note On messages, Running Status may also be used for all Mode and Control Change messages. Running Status can drastically reduce the amount of data sent by Continuous Controllers.

In some cases, the receiver must keep the status byte of the mode messages in a Running Status buffer even though the mode message is designated for a channel other than the receiver's basic channel. For example, if an Omni Off mode message is sent followed by Running Status Control Change messages, the Control Change messages can be properly recognized even though the Omni Off message may have been ignored.

Running Status Buffer and Response of Receiver to Different Messages
(Basic Channel = 3, Mode: Omni On)



The receiver should always hold the last status byte in a Running Status buffer in case the transmitter is utilizing Running Status to reduce the number of bytes sent. This also means the receiver has to determine how many data bytes (one or two) are associated with each message. It is recommended that the Running Status buffer be set up as follows:

1. Buffer is cleared at power up.
2. Buffer stores the status when a channel message is received.
3. Buffer is cleared when a System Exclusive or Common status message is received.
4. Nothing is done to the buffer during reception of real time messages.
5. The data bytes are ignored when the value of the status buffer is 0 (zero).

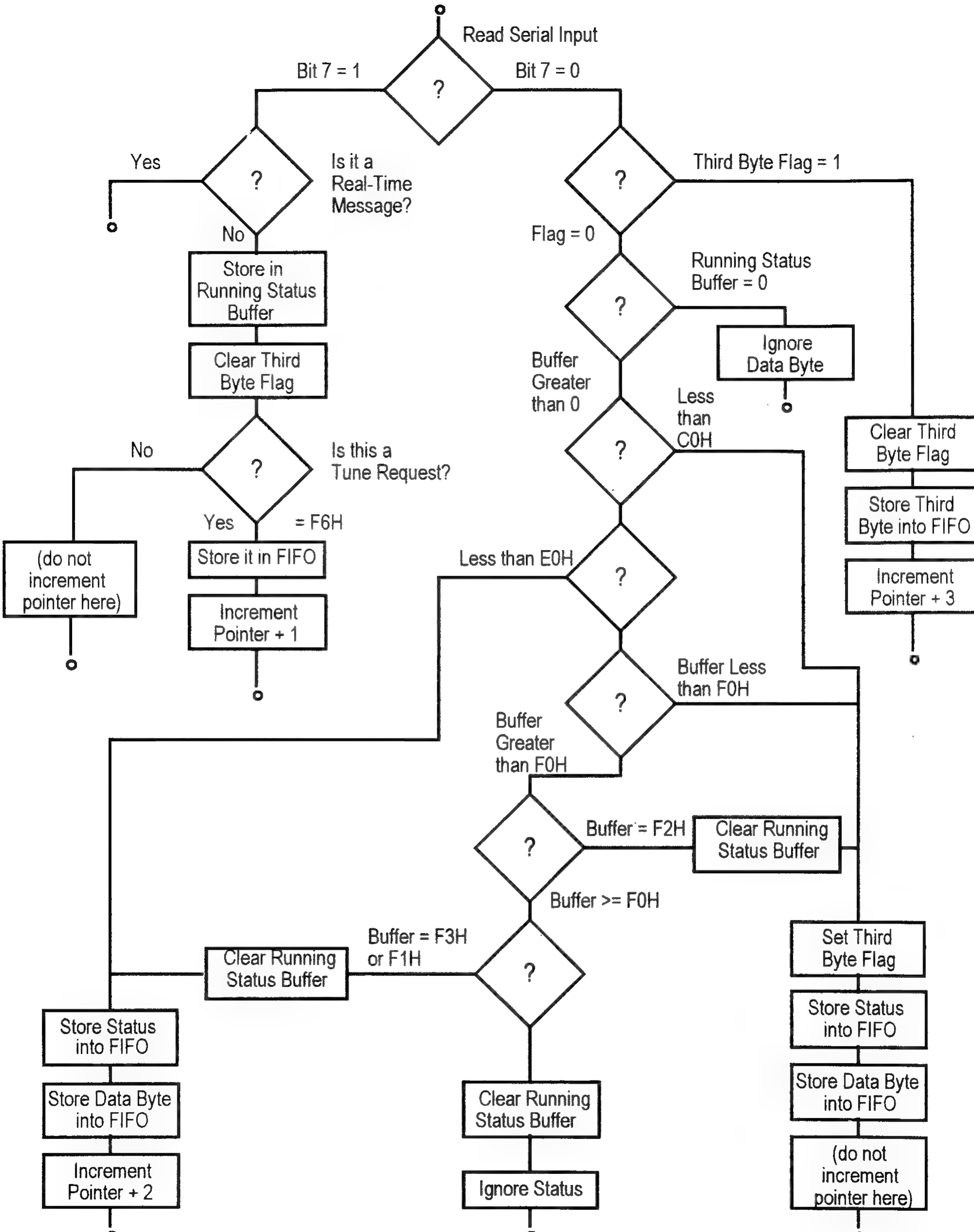
There are currently two undefined System Common status bytes (F4H and F5H). Should one of these undefined messages be received, it should be ignored and the running status buffer should be cleared. There are currently two undefined Real Time status bytes (F9H, FDH). Since these may convey only timing information, they should always be ignored, and the running status buffer should remain unaffected.

If Running Status is being used and a receiver is connected to a transmitter after the transmitter has powered on it will not play until the next Status byte is transmitted. For this reason it is recommended that the status be refreshed every few seconds.

To Summarize:

A transmitter may or may not be programmed to take advantage of Running Status. Using Running Status, notes may be turned off by sending a Note On message with zero velocity . It is the responsibility of the receiver to always recognize both normal and running status modes.

A receiver should take into consideration that a transmitter can send messages in either Running Status or normal modes. The following flowchart shows an example of an interrupt-driven routine:


Read Serial Input
Bit 7 = 1
Bit 7 = 0
?
Yes
Is it a Real-Time Message?
No
Store in Running Status Buffer
Clear Third Byte Flag
No
Is this a Tune Request?
Yes
= F6H
(do not increment pointer here)
Store it in FIFO
Increment Pointer + 1
Third Byte Flag = 1
Flag = 0
Running Status Buffer = 0
Ignore Data Byte
Buffer Greater than 0
Less than C0H
Clear Third Byte Flag
Store Third Byte into FIFO
Increment Pointer + 3
Less than E0H
Buffer Less than F0H
Buffer Greater than F0H
Buffer = F2H
Clear Running Status Buffer
Buffer >= F0H
Buffer = F3H or F1H
Clear Running Status Buffer
Store Status into FIFO
Store Data Byte into FIFO
Increment Pointer + 2
Clear Running Status Buffer
Ignore Status
Set Third Byte Flag
Store Status into FIFO
Store Data Byte into FIFO
(do not increment pointer here)

ASSIGNMENT OF NOTE ON/OFF COMMANDS

If an instrument receives two or more Note On messages with the same key number and MIDI channel. it must make a determination of how to handle the additional Note Ons. It is up to the receiver as to whether the same voice or another voice will be sounded. or if the messages will be ignored. The transmitter. however. must send a corresponding Note Off message for every Note On sent. If the transmitter were to send only one Note Off message, and if the receiver in fact assigned the two Note On messages to different voices. then one note would linger. Since there is no harm or negative side effect in sending redundant Note Off messages this is the recommended practice.

VOICE ASSIGNMENT IN POLY MODE

In Poly mode there are no particular rules which define how to assign voices when more than one Note On message is received and recognized. If more Note On messages are transmitted than the receiver is capable of playing. the receiver is free to use any method of dealing with this "overflow" situation (such as first vs. last note priority). The priority of voice assignments may follow the order in which Note On messages are received. the receiver's own keyboard control logic. or some other scheme.

When a transmitter sends Note On and Off information to a slave keyboard which is also being played. it is important for the receiver to distinguish the source of Note On/Off information. For example. a Note Off received from MIDI should not turn off a note that is being played on the slave keyboard. Conversely. releasing a key on the slave's keyboard should not turn off a note being received from MIDI.

"ALL NOTES OFF" FUNCTION WHEN SWITCHING MODES

When a receiver is switching between Omni On/Off and Poly or Mono modes. all notes should be turned off . This is to avoid any unexpected behavior when the instrument's mode is switched. Caution should be taken to turn off only those note events received from MIDI and not those played on the receiver's keyboard.

MIDI MERGING AND ALL NOTES OFF

A sequencer replays previously recorded messages and merges them with any messages received at its MIDI In. A MIDI merging device combines two incoming data streams in real time. In either case the result is that a single MIDI data stream is communicating information produced by more than one transmitter. If an All Notes Off messages is passed through either a sequencer or merging device, then all connected devices will shut off their notes, though the All Notes Off may have only been intended for the notes turned on by one specific instrument. When an All Notes Off is received by a sequencer it should check to make sure that it does not conflict with any notes currently being played. Similarly. if an All Notes Off message is contained in the recorded sequence, it should not be sent if Note On data for that channel is being received. A MIDI merging device should feature the ability to selectively filter All Notes Off messages to avoid this problem.

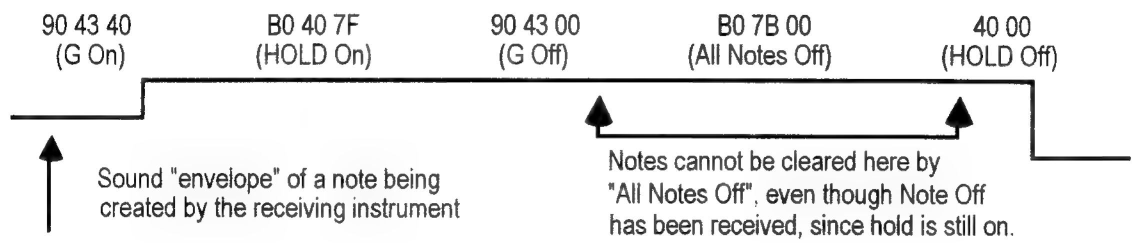
Mode messages with a second byte greater than 124 should be treated in the same way as the All Notes Off message since they also perform an All Notes Off function.

THE RELATIONSHIP BETWEEN THE HOLD PEDAL AND "ALL NOTES OFF"

If Note Off messages are received while the hold pedal (controller 64 (40H)) is 'on' they must be recognized but not acted upon until the release of the hold pedal. The same is true for the All Notes Off message. A Hold or Sustain pedal 'On' message takes priority over Note Off and All Notes Off until it is released.

All Notes Off should force voices to go to the release stage of the envelope, and not terminate the sound of the notes abruptly.

MIDI Messages Transmitted:



FURTHER DESCRIPTION OF HOLD PEDAL

Hold and Second Release switches use controller number 64 (40H). Proper implementation of the hold pedal will maintain the envelope's sustain level. A "Hold 2" switch has been defined as controller number 69 (45H) as a means of implementing all other hold functions (e.g. "freeze" where envelopes etc. are frozen in their current state) and/or for implementing two different hold functions simultaneously. "Chord Hold" which holds only the notes held when the foot pedal is switched on, is equivalent to the Sostenuto controller 66. All notes played after the foot pedal is switched on are performed normally.

PRIORITY OF MIDI RECEIVING

An instrument capable of receiving and processing incoming MIDI data must give priority to its MIDI In port over its local functions such as the front panel or keyboard. It is critical that incoming data never be ignored or mishandled due to the processor's attention being elsewhere.

At 31.25 Kbaud, one byte is sent every 320 microseconds, which means that proper handling of the received data during any long-term or ongoing MIDI communication will require a high speed microprocessor. For this reason, interrupts and FIFO (first in/first out) buffers are commonly used. As soon as possible after an interrupt is generated, received data can be stored in a FIFO buffer for processing later on. This data handling can take much less than 320 µS, allowing time for the microprocessor to handle other aspects of the instrument's operation.

RELEASE OF OMNI

As a transmitter has no way of knowing what channel a receiver is on it is best to always be able to turn Omni off by means of front panel controls on an instrument.

BASIC CHANNEL OF A SEQUENCER

To a receiver, the output of a MIDI sequencer is identical to the output of any MIDI transmitter with the possible exception of added Real Time bytes. A transmitting instrument sends on a particular channel which a sequencer then records and re-transmits. Thus, a sequencer does not need a Basic Channel as do other instruments. However, this does not prevent a sequencer from having a Basic Channel for recognizing mode messages or changing channels.

TRANSPOSING

If key transpose is implemented on a keyboard instrument, the MIDI key number 60 can be assigned to a physical key other than middle C. Transposition is allowed on both transmitters and receivers. The transposing system in a device should separately affect the keyboard data and incoming MIDI data going to the voice module. To avoid confusion it is a good idea to use an indicator to show when key transpose is active.

MIDI IMPLEMENTATION CHART INSTRUCTIONS

The standard MIDI Implementation Chart is used as a quick reference of transmitter and receiver functions so that users can easily recognize what messages and functions are implemented in the instrument. This chart should be included in the users manual of all MIDI products. For example, if a user intends to connect two MIDI instruments, they might compare the "Transmitted" part of one instrument's chart, with the "Recognized" part of the other instrument's chart by overlapping them. For this reason each chart should be the same size and have the same number of lines.

GENERAL

1. The "[ ]" brackets at the top of the chart is used for the instrument's name such as, [LINEAR WAVETABLE SYNTHESIZER].

2. The item "MODEL" should be used for the model number, such as "LW-1".

3. The body of the implementation chart is divided into four columns. The first column is the specific function or item, the next two columns give information on whether the specified function is transmitted and/or received, and the fourth column is used for remarks. This last column is useful to explain anything unique to this implementation.

FUNCTION DESCRIPTION

1. BASIC CHANNEL:

   Default: Channel which is assigned when power is first applied to unit.
   Changed: The channels which can be assigned from the instrument's front panel.

2. MODE:

   Default: This is the channel mode used by a Transmitter and Receiver when power is first applied. This should be written as Mode x (where x is 1 through 4), as shown on the bottom of sheet.
   Messages: These are the mode messages which can be transmitted or received, such as OMNI ON/OFF, MONO ON, and POLY ON. MONO ON and POLY ON may be written in the short form "MONO", "POLY".
   Altered: This shows the channel modes which are not implemented by a receiver and the modes which are substituted. For example, if the receiver cannot accept "MONO ON" mode, but will switch to "OMNI ON" mode in order to receive the MIDI data, the following expression should be used: "MONO ON > OMNI ON" or "MONO > OMNI".

3. NOTE NUMBER:

   Note Number: The total range of transmitted or recognized notes.
   True Voice: Range of received note numbers falling within the range of true notes produced by the instrument.

4. VELOCITY:

NOTE ON/NOTE OFF Velocity is assigned either an "o" for implemented or an "x" for not implemented. In the space following the "o" or "x" it may be mentioned how the Note On or Off data is being transmitted.

5. AFTERTOUCH:

"o" for implemented or an "x" for not implemented

6. PITCH BEND:

"o" for implemented or an "x" for not implemented

7. CONTROL CHANGE:

Space is given in this area for listing of any implemented control numbers. An "o" or "x" should be placed in the appropriate Transmitted or Recognized column and the function of the specified control number should be listed in the remarks column.

8. PROGRAM CHANGE:

"o" for implemented or an "x" for not implemented. If implemented, the range of numbers should be included.

True # (Number): The range of the program change numbers which correspond to the actual number of patches selected.

9. SYSTEM EXCLUSIVE:

"o" for implemented or an "x" for not implemented. A full description of the instrument's System Exclusive implementation should be included on separate pages.

10. SYSTEM COMMON:

"o" for implemented or an "x" for not implemented. The following abbreviations are used:

Song Pos = Song Position
Song Sel = Song Select
Tune = Tune Request

11. SYSTEM REAL TIME:

"o" for implemented or an "x" for not implemented. The following abbreviations are used:

Clock = Timing Clock
Commands = Start, Continue and Stop

12. AUX. MESSAGES:

"o" for implemented or an "x" for not implemented. The following abbreviations are used:

Aux = Auxiliary
Active Sense = Active Sensing

13. NOTES:

The "Notes" column can be any comments such as:

Power Up messages transmitted, implementation of program changes to additional memory banks, etc.

Model MIDI Implementation Chart

Date:
Version:

Function...		Transmitted	Recognized	Remarks
Basic Channel	Default Changed			
Mode	Default Messages Altered	**************		
Note Number	True voice	**************		
Velocity	Note On Note Off			
After Touch	Key's Channel			
Pitch Bend				
Control Change				
Program Change	True Number	**************		
System Exclusive				
System Common	Song Position Song Select Tune Request			
System Real Time	Clock Commands			
Aux Messages	Local On/Off All Notes Off Active Sensing System Reset			
Notes				

Mode 1: Omni On, Poly Mode 2: Omni On, Mono O: Yes
Mode 3: Omni Off, Poly Mode 4: Omni Off, Mono X: No

TABLE I

SUMMARY OF STATUS BYTES

Hex	STATUS Binary D7--D0		NUMBER OF DATA BYTES	DESCRIPTION
Channel Voice Messages				
8nH	1000nnnn		2	Note Off
9nH	1001nnnn		2	Note On (a velocity of 0 = Note Off)
AnH	1010nnnn		2	Polyphonic key pressure/Aftertouch
BnH	1011nnnn		2	Control change
CnH	1100nnnn		1	Program change
DnH	1101nnnn		1	Channel pressure/After touch
EnH	1110nnnn		2	Pitch bend change
Channel Mode Messages				
BnH	1011nnnn	(01111xxx)	2	Selects Channel Mode
System Messages				
F0H	11110000		*****	System Exclusive
	11110sss		0 to 2	System Common
	11111ttt		0	System Real Time

NOTES:

nnnn: N-1, where N = Channel #,
i.e. 0000 is Channel 1, 0001 is Channel 2,
and 1111 is Channel 16.

*****: 0iiiiiii, data, ..., EOX
iiiiiii: Identification
sss: 1 to 7
ttt: 0 to 7
xxx: Channel Mode messages are sent under the same Status Byte as the Control Change messages (BnH). They are differentiated by the first data byte which will have a value from 121 to 127 for Channel Mode messages.

TABLE II

CHANNEL VOICE MESSAGES

STATUS Hex	Binary	DATA BYTES	DESCRIPTION
8nH	1000nnnn	0kkkkkkk 0vvvvvvv	Note Off vvvvvvv: note off velocity
9nH	1001nnnn	0kkkkkkk 0vvvvvvv	Note On vvvvvvv ≠ 0: velocity vvvvvvv = 0: note off
AnH	1010nnnn	0kkkkkkk 0vvvvvvv	Polyphonic Key Pressure (Aftertouch) vvvvvvv: pressure value
BnH	1011nnnn	0ccccccc 0vvvvvvv	Control Change (See Table III) ccccccc: control # (0-119) vvvvvvv: control value ccccccc = 120 thru 127: Reserved. (See Table IV)
CnH	1100nnnn	0ppppppp	Program Change ppppppp: program number (0-127)
DnH	1101nnnn	0vvvvvvv	Channel Pressure (Aftertouch) vvvvvvv: pressure value
EnH	1110nnnn	0vvvvvvv 0vvvvvvv	Pitch Bend Change LSB Pitch Bend Change MSB

NOTES:

1. nnnn: Voice Channel number (1-16, coded as defined in Table I notes)

2. kkkkkkk: note number (0 - 127)

3. vvvvvvv: key velocity
   A logarithmic scale is recommended.

4. Continuous controllers are divided into Most Significant and Least Significant Bytes. If only seven bits of resolution are needed for any particular controllers, only the MSB is sent. It is not necessary to send the LSB. If more resolution is needed, then both are sent, first the MSB, then the LSB. If only the LSB has changed in value, the LSB may be sent without re-sending the MSB.

TABLE III

CONTROLLER NUMBERS

CONTROL NUMBER (2nd Byte value) Decimal	Hex	CONTROL FUNCTION
0	00H	Bank Select
1	01H	Modulation wheel or lever
2	02H	Breath Controller
3	03H	Undefined
4	04H	Foot controller
5	05H	Portamento time
6	06H	Data entry MSB
7	07H	Channel Volume (formerly Main Volume)
8	08H	Balance
9	09H	Undefined
10	0AH	Pan
11	0BH	Expression Controller
12	0CH	Effect Control 1
13	0DH	Effect Control 2
14-15	0E-0FH	Undefined
16-19	10-13H	General Purpose Controllers (#'s 1-4)
20-31	14-1FH	Undefined
32-63	20-3FH	LSB for values 0-31
64	40H	Damper pedal (sustain)
65	41H	Portamento On/Off
66	42H	Sostenuto
67	43H	Soft pedal
68	44H	Legato Footswitch (vv = 00-3F:Normal, 40-7F=Legatto)
69	45H	Hold 2
70	46H	Sound Controller 1 (default: Sound Variation)
71	47H	Sound Controller 2 (default: Timbre/Harmonic Intensity)
72	48H	Sound Controller 3 (default: Release Time)
73	49H	Sound Controller 4 (default: Attack Time)
74	4AH	Sound Controller 5 (default: Brightness)
75-79	4BH-4FH	Sound Controllers 6-10 (no defaults)
80-83	50-53H	General Purpose Controllers (#'s 5-8)
84	54H	Portamento Control
85-90	55-5AH	Undefined
91	5BH	Effects 1 Depth (formerly External Effects Depth)
92	5CH	Effects 2 Depth (formerly Tremolo Depth)
93	5DH	Effects 3 Depth (formerly Chorus Depth)
94	5EH	Effects 4 Depth (formerly Celeste (Detune) Depth)
95	5FH	Effects 5 Depth (formerly Phaser Depth)
96	60H	Data increment
97	61H	Data decrement
98	62H	Non-Registered Parameter Number LSB
99	63H	Non-Registered Parameter Number MSB
100	64H	Registered Parameter Number LSB
101	65H	Registered Parameter Number MSB
102-119	66-77H	Undefined
120-127	78-7FH	Reserved for Channel Mode Messages

TABLE IIIa

REGISTERED PARAMETER NUMBERS

Parameter Number LSB	MSB	Function
00H	00H	Pitch Bend Sensitivity
01H	00H	Fine Tuning
02H	00H	Coarse Tuning
03H	00H	Tuning Program Select
04H	00H	Tuning Bank Select

TABLE IV

CHANNEL MODE MESSAGES

STATUS Hex	Binary	DATA BYTES	DESCRIPTION
Bn	1011nnnn	0ccccccc 0vvvvvvv	Mode Messages ccccccc = 120: All Sound Off vvvvvvv = 0 ccccccc = 121: Reset All Controllers vvvvvvv = 0 ccccccc = 122:Local Control vvvvvvv = 0, Local Control Off vvvvvvv = 127, Local Control On ccccccc = 123: All Notes Off vvvvvvv = 0 ccccccc = 124: Omni Mode Off (All Notes Off) vvvvvvv = 0 ccccccc = 125:Omni Mode On (All Notes Off) vvvvvvv = 0 ccccccc = 126: Mono Mode On (Poly Mode Off) (All Notes Off) vvvvvvv = M, where M is the number of channels. vvvvvvv = 0, the number of channels equals the number of voices in the receiver. ccccccc = 127: Poly Mode On (Mono Mode Off) (All Notes Off) vvvvvvv = 0

NOTES:

1. nnnn: Basic Channel number (1-16)

2. ccccccc: Controller number (121 - 127)

3. vvvvvvv: Controller value

TABLE V

SYSTEM COMMON MESSAGES

STATUS Hex	Binary	DATA BYTES	DESCRIPTION
F1H	11110001	0nnndddd	MIDI Time Code Quarter Frame nnn: Message Type dddd: Values
F2H	11110010	 0lllllll 0hhhhhhh	Song Position Pointer lllllll: (Least significant) hhhhhhh: (Most significant)
F3H	11110011	0sssssss	Song Select sssssss: Song #
F4H	11110100		Undefined
F5H	11110101		Undefined
F6H	11110110	none	Tune Request
F7H	11110111	none	EOX: "End of System Exclusive" flag

TABLE VI

SYSTEM REAL TIME MESSAGES

STATUS Hex	Binary	DATA BYTES	DESCRIPTION
F8H	11111000		Timing Clock
F9H	11111001		Undefined
FAH	11111010		Start
FBH	11111011		Continue
FCH	11111100		Stop
FDH	11111101		Undefined
FEH	11111110		Active Sensing
FFH	11111111		System Reset

TABLE VII

SYSTEM EXCLUSIVE MESSAGES

STATUS Hex	Binary	DATA BYTES	DESCRIPTION
F0H	11110000		Bulk dump, etc.
		0iiiiiii	iiiiiii: identification *(See note 1)*
		.	
		(0ddddddd)	
		.	Any number of data bytes may be sent here, for any purpose,
		.	as long as they all have a zero in the most significant bit.
		.	
		(0ddddddd)	
F7H	11110111		EOX: "End of System Exclusive"

NOTES:

1. iiiiiii: identification ID (0-127).

   A) If the ID is 0, the following two bytes are used as extensions to the manufacturer ID.
   B) A manufacturer's ID number can be obtained from the MMA, except for Japan.
   C) ID 7DH is reserved for non-commercial use (e.g. schools, research, etc.) and is not to be used on any product released to the public; 7EH (Non-Real Time) and 7FH (Real Time) are used for extensions to the MIDI specification

2. All bytes between the System Exclusive Status byte and EOX must have zeroes in the MSB.

3. Status or Data bytes (except System Real-Time) should not be interleaved with System Exclusive

4. No Status Bytes (other than System Real-Time) should be sent until after an EOX has terminated the System Exclusive message. If however, a Status Byte other than EOX is received during a System Exclusive message, the message is terminated.

TABLE VIIa

CURRENTLY DEFINED UNIVERSAL SYSTEM EXCLUSIVE MESSAGES

	SUB-ID #1	SUB-ID #2	DESCRIPTION
Non-Real Time (7EH)			
	00	--	**Unused**
	01	(not used)	**Sample Dump Header**
	02	(not used)	**Sample Data Packet**
	03	(not used)	**Sample Dump Request**
	04	nn	**MIDI Time Code**
		00	Special
		01	Punch In Points
		02	Punch Out Points
		03	Delete Punch In Point
		04	Delete Punch Out Point
		05	Event Start Point
		06	Event Stop Point
		07	Event Start Points with additional info.
		08	Event Stop Points with additional info.
		09	Delete Event Start Point
		0A	Delete Event Stop Point
		0B	Cue Points
		0C	Cue Points with additional info.
		0D	Delete Cue Point
		0E	Event Name in additional info.
	05	nn	**Sample Dump Extensions**
		01	Multiple Loop Points
		02	Loop Points Request
	06	nn	**General Information**
		01	Identity Request
		02	Identity Reply
	07	nn	**File Dump**
		01	Header
		02	Data Packet
		03	Request
	08	nn	**MIDI Tuning Standard**
		00	Bulk Dump Request
		01	Bulk Dump Reply
	09	nn	**General MIDI**
		01	General MIDI System On
		02	General MIDI System Off
	7B	(not used)	**End Of File**
	7C	(not used)	**Wait**
	7D	(not used)	**Cancel**
	7E	(not used)	**NAK**
	7F	(not used)	**ACK**

Real Time (7FH)

00	--	**Unused**
01	nn	**MIDI Time Code**
	01	Full Message
	02	User Bits
02	nn	**MIDI Show Control**
	00	MSC Extensions
	01 - 7F	MSC Commands
		(Detailed in MSC documentation)
03	nn	**Notation Information**
	01	Bar Number
	02	Time Signature (Immediate)
	42	Time Signature (Delayed)
04	nn	**Device Control**
	01	Master Volume
	02	Master Balance
05	nn	**Real Time MTC Cueing**
	00	Special
	01	Punch In Points
	02	Punch Out Points
	03	(Reserved)
	04	(Reserved)
	05	Event Start points
	06	Event Stop points
	07	Event Start points with additional info.
	08	Event Stop points with additional info.
	09	(Reserved)
	0A	(Reserved)
	0B	Cue points
	0C	Cue points with additional info.
	0D	(Reserved)
	0E	Event Name in additional into.
06	nn	**MIDI Machine Control Commands**
	00 - 7F	MMC Commands
		(Detailed in MMC documentation)
07	nn	**MIDI Machine Control Responses**
	00 - 7F	MMC Commands
		(Detailed in MMC documentation)
08	nn	**MIDI Tuning Standard**
	02	Note Change

NOTES:

1. The standardized format for both Real Time and Non-Real Time messages is as follows:
   `F0H <ID number> <device ID> <sub-ID#1> <sub-ID#2>...... F7H`

2. Additional details and descriptions of MTC MSC and MMC are available as separate documents.

TABLE VIIb

SYSTEM EXCLUSIVE MANUFACTURER'S ID NUMBERS

NUMBER	MANUFACTURER	NUMBER	MANUFACTURER
American Group			
01H	Sequential	00H 00H 1FH	Zeta Systems
02H	IDP	00H 00H 20H	Axxes
03H	Voyetra/Octave-Plateau	00H 00H 21H	Orban
04H	Moog	00H 00H 24H	KTI
05H	Passport Designs	00H 00H 25H	Breakaway Technologies
06H	Lexicon	00H 00H 26H	CAE
07H	Kurzweil	00H 00H 29H	Rocktron Corp.
08H	Fender	00H 00H 2AH	PianoDisc
09H	Gulbransen	00H 00H 2BH	Cannon Research Group
0AH	AKG Acoustics	00H 00H 2DH	Rogers Instrument Corp.
0BH	Voyce Music	00H 00H 2EH	Blue Sky Logic
0CH	Waveframe Corp	00H 00H 2FH	Encore Electronics
0DH	ADA Signal Processors	00H 00H 30H	Uptown
0EH	Garfield Electronics	00H 00H 31H	Voce
0FH	Ensoniq	00H 00H 32H	CTI Audio, Inc. (Music. Intel Dev.)
10H	Oberheim	00H 00H 33H	S&S Research
11H	Apple Computer	00H 00H 34H	Broderbund Software, Inc.
12H	Grey Matter Response	00H 00H 35H	Allen Organ Co.
13H	Digidesign	00H 00H 37H	Music Quest
14H	Palm Tree Instruments	00H 00H 38H	APHEX
15H	JLCooper Electronics	00H 00H 39H	Gallien Krueger
16H	Lowrey	00H 00H 3AH	IBM
17H	Adams-Smith	00H 00H 3CH	Hotz Instruments Technologies
18H	Emu Systems	00H 00H 3DH	ETA Lighting
19H	Harmony Systems	00H 00H 3EH	NSI Corporation
1AH	ART	00H 00H 3FH	Ad Lib, Inc.
1BH	Baldwin	00H 00H 40H	Richmond Sound Design
1CH	Eventide	00H 00H 41H	Microsoft
1DH	Inventronics	00H 00H 42H	The Software Toolworks
1FH	Clarity	00H 00H 43H	Niche/RJMG
00H 00H 01H	Time Warner Interactive	00H 00H 44H	Intone
00H 00H 07H	Digital Music Corp.	00H 00H 47H	GT Electronics/Groove Tubes
00H 00H 08H	IOTA Systems	00H 00H 4FH	InterMIDI, Inc.
00H 00H 09H	New England Digital	00H 00H 49H	Timeline Vista
00H 00H 0AH	Artisyn	00H 00H 4AH	Mesa Boogie
00H 00H 0BH	IVL Technologies	00H 00H 4CH	Sequoia Development
00H 00H 0CH	Southern Music Systems	00H 00H 4DH	Studio Electronics
00H 00H 0DH	Lake Butler Sound Company	00H 00H 4EH	Euphonix
00H 00H 0EH	Alesis	00H 00H 4FH	InterMIDI
00H 00H 10H	DOD Electronics	00H 00H 50H	MIDI Solutions
00H 00H 11H	Studer-Editech	00H 00H 51H	3DO Company
00H 00H 14H	Perfect Fretworks	00H 00H 52H	Lightwave Research
00H 00H 15H	KAT	00H 00H 53H	Micro-W
00H 00H 16H	Opcode	OOH OOH 54H	Spectral Synthesis
00H 00H 17H	Rane Corp.	OOH OOH 55H	Lone Wolf
00H 00H 18H	Anadi Inc.	00H 00H 56H	Studio Technologies
00H 00H 19H	KMX	00H 00H 57H	Peterson EMP
00H 00H 1AH	Allen & Heath Brenell	00H 00H 58H	Atari
00H 00H 1BH	Peavey Electronics	00H 00H 59H	Marion Systems
00H 00H 1CH	360 Systems	00H 00H 5AH	Design Event
00H 00H 1DH	Spectrum Design and Development	00H 00H 5BH	Winjammer Software
00H 00H 1EH	Marquis Music	00H 00H 5CH	AT&T Bell Labs

NUMBER	MANUFACTURER
00H 00H 5EH	Symetrix
00H 00H 5FH	MIDI the World
00H 00H 60H	Desper Products
00H 00H 61H	Micros 'N MIDI
00H 00H 62H	Accordians Intl
00H 00H 63H	EuPhonics
00H 00H 64H	Musonix
00H 00H 65H	Turtle Beach Systems
00H 00H 66H	Mackie Designs
00H 00H 67H	Compuserve
00H 00H 68H	BES Technologies
00H 00H 69H	QRS Music Rolls
00H 00H 6AH	P G Music
00H 00H 6BH	Sierra Semiconductor
00H 00H 6CH	EpiGraf Audio Visual
00H 00H 6DH	Electronics Deiversified
00H 00H 6EH	Tune 1000
00H 00H 6FH	Advanced Micro Devices
00H 00H 70H	Mediamation
00H 00H 71H	Sabine Music
00H 00H 72H	Woog Labs
00H 00H 73H	Micropolis
00H 00H 74H	Ta Horng Musical Inst.
00H 00H 75H	eTek (formerly Forte)
00H 00H 76H	Electrovoice
00H 00H 77H	Midisoft
00H 00H 78H	Q-Sound Labs
00H 00H 79H	Westrex
00H 00H 7AH	NVidia
00H 00H 7BH	ESS Technology
00H 00H 7CH	MediaTrix Peripherals
00H 00H 7DH	Brooktree
00H 00H 7EH	Otari
00H 00H 7FH	Key Electronics
00H 01H 01H	Crystalake Multimedia
00H 01H 02H	Crystal Semiconductor
00H 01H 03H	Rockwell Semiconductor

European Group

NUMBER	MANUFACTURER
20H	Passac
21H	SIEL
22H	Synthaxe
24H	Hohner
25H	Twister
26H	Solton
27H	Jellinghaus MS
28H	Southworth Music Systems
29H	PPG
2AH	JEN
2BH	SSL Limited
2CH	Audio Veritrieb
2FH	Elka
30H	Dynacord
31H	Viscount
33H	Clavia Digital Instruments
34H	Audio Architecture
35H	GeneralMusic Corp.
39H	Soundcraft Electronics
3BH	Wersi
3CH	Avab Electronik Ab
3DH	Digigram
3EH	Waldorf Electronics
3FH	Quasimidi
00H 20H 00H	Dream
00H 20H 01H	Strand Lighting
00H 20H 02H	Amek Systems
00H 20H 04H	Böhm Electronic
00H 20H 06H	Trident Audio
00H 20H 07H	Real World Studio
00H 20H 09H	Yes Technology
00H 20H 0AH	Audiomatica
00H 20H 0BH	Bontempi/Farfisa
00H 20H 0CH	F.B.T. Elettronica
00H 20H 0DH	MidiTemp
00H 20H 0EH	LA Audio (Larking Audio)
00H 20H 0FH	Zero 88 Lighting Limited
00H 20H 10H	Micon Audio Electronics GmbH
00H 20H 11H	Forefront Technology
00H 20H 13H	Kenton Electronics
00H 20H 15H	ADB
00H 20H 16H	Marshall Products
00H 20H 17H	DDA
00H 20H 18H	BSS
00H 20H 19H	MA Lighting Technology
00H 20H 1AH	Fatar
00H 20H 1BH	QSC Audio
00H 20H 1CH	Artisan Classic Organ
00H 20H 1DH	Orla Spa
00H 20H 1EH	Pinnacle Audio
00H 20H 1FH	TC Electonics
00H 20H 20H	Doepfer Musikelektronik
00H 20H 21H	Creative Technology Pte
00H 20H 22H	Minami/Seiyddo
00H 20H 23H	Goldstar
00H 20H 24H	Midisoft s.a.s di M. Cima
00H 20H 25H	Samick
00H 20H 26H	Penny and Giles
00H 20H 27H	Acorn Computer
00H 20H 28H	LSC Electronics
00H 20H 29H	Novation EMS
00H 20H 2AH	Samkyung Mechatronics
00H 20H 2BH	Medeli Electronics
00H 20H 2CH	Charlie Lab
00H 20H 2DH	Blue Chip Music Tech
00H 20H 2EH	BEE OH Corp

NUMBER	MANUFACTURER	NUMBER	MANUFACTURER

Japanese Group (as of 10/92)

40H	Kawai
41H	Roland
42H	Korg
43H	Yamaha
44H	Casio
46H	Kamiya Studio
47H	Akai
48H	Japan Victor
49H	Mesosha
4AH	Hoshino Gakki
4BH	Fujitsu Elect
4CH	Sony
4DH	Nisshin Onpa
4EH	TEAC
50H	Matsushita Electric
51H	Fostex
52H	Zoom
53H	Midori Electronics
54H	Matsushita Communication Industrial
55H	Suzuki Musical Inst. Mfg.

TABLE VIII

ADDITIONAL OFFICIAL SPECIFICATION DOCUMENTS PUBLISHED BY
THE MIDI MANUFACTURERS ASSCOCIATION

DOCUMENT TITLE	DESCRIPTION
MIDI Time Code	Recommended Practice RP004/RP008
MIDI Show Control 1.0	Recommended Practice RP002
MIDI Machine Control	Recommended Practice RP013
Standard MIDI Files	Recommended Practice RP001
General MIDI System Level 1	Recommended Practice RP003